SISTEM PENJADWALAN DOKTER JAGA MENGGUNAKAN ALGORITMA GREEDY DENGAN PERMUTASI (STUDI KASUS : RSUD ARIFIN ACHMAD PEKANBARU)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh :

POPI SELPIRA
10551001490

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2010

SISTEM PENJADWALAN DOKTER JAGA MENGGUNAKAN ALGORITMA GREEDY DENGAN PERMUTASI (STUDI KASUS : RSUD ARIFIN ACHMAD PEKANBARU)

POPI SELPIRA
10551001490

Tanggal Sidang : 22 Juni 2010
Periode Wisuda : Juli 2010

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Penjadwalan dokter jaga merupakan salah satu persoalan yang dihadapi oleh pihak RSUD Arifin Achmad Pekanbaru. Permasalahan yang dihadapi adalah ketika melakukan penjadwalan, terlebih dahulu harus diperiksa satu per satu *shift* yang masih kosong dan disesuaikan dengan jadwal dokter yang bersangkutan. Jika terdapat *request shift* jaga, maka pihak yang menyusun jadwal harus memeriksa kembali apakah *shift* tersebut telah diisi atau belum oleh dokter lain. Hal ini membutuhkan waktu yang relatif lebih lama terutama bila jadwal yang disusun cukup banyak.

Pada tugas akhir ini dirancang bangun sebuah sistem penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi. *Greedy* merupakan algoritma yang praktis, ringkas dan *fleksible*, sehingga sesuai untuk diimplementasikan pada berbagai kasus persoalan termasuk dalam penjadwalan dokter jaga. Permutasi digunakan agar dapat mengarahkan algoritma *greedy* yang dibuat. Sistem ini dirancang menggunakan *Microsoft Visual Basic 6.0 dan Microsoft Access 2003*. Sistem penjadwalan dokter jaga dibagi atas tiga tahapan, yaitu; penentuan *constraints*, penghitungan *shift* dan penugasan dokter. Sistem ini menghasilkan jadwal jaga yang sesuai dengan prosedur jadwal jaga yang terdapat di RSUD Arifin Achmad Pekanbaru, dimana hasilnya adalah berupa laporan jadwal jaga yang berbentuk *softfile* dalam format *Ms Word*, *Ms Excel* dan *Acrobat Format* (PDF).

Kata Kunci : Algoritma *Greedy*, Penjadwalan Dokter Jaga, Permutasi.

SISTEM PENJADWALAN DOKTER JAGA MENGGUNAKAN ALGORITMA GREEDY DENGAN PERMUTASI (STUDI KASUS : RSUD ARIFIN ACHMAD PEKANBARU)

POPI SELPIRA
10551001490

Date of Final Exam : June 22th 2010
Graduation Cremony Priod : July, 2010

Informatics Engineering Departement
Faculty of Sciences and Technology
State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

The Scheduling of doctors is one of the problems faced by the RSUD Arifin Achmad Pekanbaru. The problem faced is that when doing the scheduling, must first be examined one by one shift which is still empty and the schedule adjusted to the relevant doctor. If there are requests shift to guard, then the parties that make up a schedule should check again whether the shift has been filled or not by another doctor. This requires a relatively longer time, especially when schedules are prepared quite a lot.

in this final project has designed an attending system scheduling of doctor that using the greedy algorithm with the permutation. Greedy represent the simple and flexible algorithm, so that it is suitable for the implementation of various problems including doctor scheduling. Permutation used to assist and instruct greedy algorithm made.This system is designed using Microsoft Visual Basic 6.0 and Microsoft Access 2003. System Scheduling of doctor is divided into three steps, that are; the determination of constraints, enumeration shift, and also doctor assignation. This system produces schedules that keep the schedule according to the procedures contained in RSUD Arifin Achmad Pekanbaru, where the result is a form of case report in the softfile in form Ms Word, Ms Excel and Acrobat form (PDF).

***Keyword** : Doctor Scheduling, Greedy Algorithm , Permutation.*

DAFTAR ISI

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penjadwalan merupakan salah satu permasalahan penting yang masih terus dibahas sampai saat ini. Hal ini dapat dibuktikan dari semakin banyak penelitian yang dilakukan berkaitan dengan penjadwalan. Beberapa contoh penelitian dalam kasus penjadwalan adalah sebagai berikut; Penjadwalan *Job Shop* Statik Dengan Algoritma *Simulated Annealing* yang menghasilkan jadwal *job shop* yang valid (Henry Pantas Panggabean, 2002), Penjadwalan Distribusi Produk Kimia Menggunakan *Coloring Graph* yang menghasilkan optimasi penjadwalan distribusi produk kimia (Dossey, 2006), Algoritma Semut Pada Penjadwalan Produksi *Job Shop* (Zainudin Zukhri, Shidiq Alhakim, 2004) dan masih banyak kasus penjadwalan lainnya.

RSUD Arifin Achmad adalah salah satu Rumah Sakit yang telah lama berdiri di kota Pekanbaru Provinsi Riau. Rumah Sakit merupakan bagian dari pelayanan kesehatan masyarakat, untuk itu Rumah Sakit ini selalu berupaya menyelenggarakan fungsi pelayanan kesehatan sesuai dengan standar internasional dan menjadi pusat rujukan bagi Rumah Sakit lainnya di Provinsi Riau. Dalam melaksanakan tugas tersebut, Rumah Sakit harus memiliki administrasi serta manajemen yang profesional. Dokter sebagai tenaga medis sangat memegang peranan penting dalam upaya pelayanan kesehatan pasien. Untuk melayani pasiennya pihak rumah sakit memiliki dokter jaga. Untuk itu

dilakukan suatu penjadwalan, agar rutinitas dokter-dokter tersebut terhadap pasiennya terkoordinir dengan baik. RSUD Arifin Achmad belum mempunyai manajemen khusus yang menangani masalah penjadwalan dokter jaga. Penjadwalan dilakukan oleh kepala instalasi atau pihak yang telah diberikan tanggung jawab untuk menyusun jadwal jaga tersebut. Penjadwalan ini meliputi tiga bagian; IRNA (Instalasi Rawat Inap), IRNA Utama (Instalasi Rawat Inap Utama), dan IRD (Instalasi Rawat Darurat). Penjadwalan dokter jaga masih dilakukan secara manual, sehingga membutuhkan waktu yang relatif lebih lama terutama bila jadwal yang disusun cukup banyak. Hal ini disebabkan, ketika penyusunan jadwal jaga diperiksa satu per satu *shift* yang masing kosong dan disesuaikan dengan jadwal *(schedule)* dokter yang bersangkutan. Jika terdapat *request* jaga, maka pihak penyusun jadwal harus mengecek ulang apakah *shift* jaga yang di *request* sudah diisi atau belum oleh dokter lain. Penjadwalan dokter jaga memiliki ketentuan-ketentuan atau aturan-aturan khusus, sehingga masalah lain muncul jika pihak yang biasanya menangani penjadwalan tersebut diganti, maka pihak yang baru ditunjuk harus mempelajari dari awal seluruh aturan atau ketentuan yang digunakan dalam penyusunan jadwal dokter jaga tersebut. Oleh karena itu dibutuhkan sebuah sistem yang dapat membantu pihak Rumah Sakit dalam melakukan penjadwalan terhadap dokter jaga dengan cara yang lebih efektif dan efisien.

Algoritma *greedy* merupakan salah satu metode yang paling populer dalam menyelesaikan kasus permasalahan. *Greedy* adalah algoritma yang praktis, ringkas dan *fleksible* sehingga dapat digunakan pada berbagai kasus persoalan.

Hal ini dibuktikan dengan terdapatnya beberapa penelitian yang menggunakan *greedy* sebagai metode dalam penyelesaian masalah, misalnya; Penjadwalan Pertandingan Liga Dengan Algoritma *Greedy* (William, 2008), Aplikasi Algoritma *Greedy* Untuk Mengoptimalkan Efisiensi Pekerjaan (Tommy Hidayat Santoso, 2008), Aplikasi Algoritma *Greedy* yang Dimodifikasi dalam Pencarian Lintasan Terpendek (Ari Wardana, 2008), dan lain-lain. Penerapan algoritma ini adalah membentuk solusi langkah per langkah *(step by step).*

Pada tugas akhir ini dirancang bangun sistem penjadwalan dokter jaga dengan menggunakan algoritma *greedy* dengan permutasi sebagai metode pengerjaannya, permutasi akan membantu mengarahkan algoritma *greedy* yang dibuat.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan pada bagian sebelumnya, maka rumusan masalah pada tugas akhir ini adalah bagaimana menerapkan algoritma *greedy* dengan permutasi untuk persoalan penjadwalan dokter jaga, sehingga dapat menghasilkan suatu sistem yang bermanfaat bagi pengguna *(user).*

1.3 Batasan Masalah

Untuk mengatasi permasalahan diatas, maka cakupan masalah akan di batasi sebagai berikut;

1. Penjadwalan dokter jaga meliputi tiga bagian, yaitu: IRNA (Instalasi Rawat Inap), IRNA Utama (Instalasi Rawat Inap Utama) dan IRD (Instalasi Rawat Darurat).

2. Metode yang digunakan dalam sistem penjadwalan dokter jaga adalah algoritma *greedy* dengan permutasi.
3. Penjadwalan dibagi atas tiga *shift,* yaitu; pagi, sore, dan malam.

1.4 Tujuan Tugas Akhir

Tujuan yang ingin dicapai dalam penulisan tugas akhir ini adalah merancang dan membangun sistem untuk kasus permasalahan penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi.

1.5 Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi 6 (enam) bab. Berikut penjelasan tentang masing-masing bab:

BAB I PENDAHULUAN

Berisi tentang deskripsi umum dari tugas akhir ini, yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penyusunan tugas akhir, serta sistematika penulisan tugas akhir.

BAB II LANDASAN TEORI

Berisi penjelasan tentang konsep penjadwalan dokter jaga di RSUD Arifin Achmad serta mengenai metode yang akan diterapkan, yaitu konsep dasar algoritma *greedy*, konsep dasar permutasi serta pengaplikasian *greedy* pada kasus penjadwalan.

BAB III METODOLOGI PENELITIAN

Metodologi penelitian adalah langkah secara sistematis dan logis yang disusun tahap demi tahap dalam pembuatan sistem. Setiap tahap saling berkesinambungan antara satu dengan yang lainnya, dimana suatu tahapan dapat dilakukan setelah tahapan sebelumnya selesai dikerjakan.

BAB IV ANALISA DAN PERANCANGAN

Bab ini akan membahas analisa dan perancangan yang meliputi deskripsi sistem, analisa sistem dan perancangan sistem.

BAB V IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dibahas mengenai sistem penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi serta pengujiannya.

BAB VI PENUTUP

Dalam bab ini akan dijelaskan mengenai kesimpulan yang didapatkan dari pembahasan tentang sistem penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi, disertai beberapa saran sebagai hasil akhir dari penelitian yang telah dilakukan.

BAB II

LANDASAN TEORI

2.1 Algoritma *Greedy*

Algoritma *greedy* merupakan salah satu metode yang paling populer dalam menyelesaikan persoalan optimasi *(Optimization Problem)*. Persoalan optimasi adalah persoalan yang menuntut pencarian solusi optimum (terbaik). Algoritma ini merupakan algoritma yang sederhana dan *fleksible* sehingga dapat digunakan pada berbagai kasus persoalan dengan hasil yang cukup memuaskan. Berikut ini akan dijelaskan mengenai definisi, konsep algoritma *greedy* serta contoh kasus penjadwalan yang menggunakan *greedy* sebagai metode penyelesaiaan.

2.1.1 Definisi Algoritma *Greedy*

Secara harfiah *greedy* berarti rakus atau tamak. Hal ini sesuai dengan cara kerjanya yang mirip dengan salah satu sifat buruk manusia yaitu rakus. *greedy algorithm* adalah kelompok algoritma yang selalu mengambil penyelesaian sementara/lokal yang terbaik dalam setiap langkahnya untuk menyelesaikan suatu permasalahan. Pilihan terbaik akan diambil pada setiap langkah tanpa perlu memikirkan bagaimana pengaruhnya terhadap penyelesaian secara keseluruhan (Hendarto, Janoe, 2007). Prinsip *greedy* adalah *"take what you can get now!"*, yaitu ambil apa yang dapat anda peroleh sekarang (Munir, Rinaldi, 2005). Tetapi tentu saja hal ini dalam konteks yang positif, sehingga metode ini banyak digunakan untuk memecahkan persoalan.

2.1.2 Konsep Dasar Algoritma *Greedy*

Berdasarkan perihal yang telah dijelaskan pada bagian sebelumnya, pendekatan dalam algoritma *greedy* adalah membuat pilihan yang kelihatannya memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal *(local optimum)* pada setiap langkah, dengan harapan akan mengarah ke solusi optimum global *(global optimum)*. Banyak algoritma yang menerapkan prinsip *greedy* ini, antara lain:

1. Algoritma Djikstra, untuk menentukan lintasan terpendek pada graf berarah maupun tak berarah.
2. Algoritma Kruskal, digunakan dalam permasalahan pedagang keliling *(Travelling Salesperson Problem)*.
3. Algoritma Prim, untuk pencarian *minimum spanning tree.*
4. Algoritma menentukan minimasi waktu dalam sistem penjadwalan *(scheduling)*.
5. Algoritma *Huffman*, digunakan untuk masalah penampatan atau kompresi data.

Komponen dari persoalan optimasi adalah:

1. Kendala *(constraints)*
2. Fungsi objektif (fungsi optimasi)

Secara umum algoritma *greedy* disusun oleh elemen-elemen berikut (Munir,Rinaldi, 2005):

1. Himpunan kandidat

   Merupakan kumpulan elemen-elemen yang akan membentuk solusi.

2. Himpunan solusi

   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.

3. Fungsi seleksi *(selection function)*

   Memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan *(feasible)*

   Memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (*constraints*) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

5. Fungsi Obyektif

   Merupakan fungsi yang meminimalkan atau memaksimalkan nilai solusi.

Dari elemen-elemen algoritma *greedy* tersebut dapat disimpulkan bahwa algoritma *greedy* melibatkan pencarian sebuah himpunan bagian/solusi (S), dari himpunan kandidat (C), dimana S harus memenuhi kriteria dari kendala *(constraints)* yang telah ditentukan. Oleh sebab itu langkah-langkah penyelesaian menggunakan algoritma *greedy* dapat diuraikan sebagai berikut:

1. Inisialisasi Solusi (S) yang kosong. Kemudian tentukan indeks (jumlah) solusi yang dibutuhkan.

2. Pilih elemen dari himpunan kandidat (C) menggunakan fungsi pemilihan. Elemen himpunan kandidat berkurang 1.
3. Uji kelayakan kandidat yang terpilih sesuai *constraint* atau kriteria yang telah ditentukan, jika layak masukkan ke S.
4. Setelah C kosong dan jika ternyata S belum terpenuhi, maka isi ulang C dan ulangi kembali proses dari tahap 2.

2.1.3 Contoh Kasus Penjadwalan Menggunakan Algoritma *Greedy*

Kasus : Penjadwalan Pertandingan Liga dengan Algoritma *Greedy* (William, 2008)

Pembahasan :

Pertandingan berbasis liga merupakan pertandingan yang dilakukan dalam satu rentang waktu tertentu, dimana dalam rentang waktu tersebut masing-masing tim diperbolehkan bertanding sebanyak 1 kali.

Terdapat 4 tim yang akan bertanding dalam kejuaraan catur, kejuaraan ini memakai sistem liga. Tim-tim tersebut diberi simbol A,B,C,D. Kejuaraan akan berlangsung selama 3 hari.

Untuk menyelesaikan persoalan ini diperlukan komponen-komponen berikut:

1. *PointString* (*Char* x, *Char* y), digunakan untuk memasukkan 1 pasang simbol tim yang akan bertanding.
2. *List of PointofChar* (S), merupakan struktur untuk jadwal pertandingan dan merupakan himpunan solusi.

3. *Array of Char* (C), Array ini merupakan himpunan seluruh peserta dan merupakan himpunan simbol kandidat.
4. Fungsi pemilihan (pilih (C))

   Fungsi ini memilih kandidat dari C sesuai kriteria pembuat algoritma. Pada kasus ini pemilihan dilakukan berdasarkan simbol terkecil.
5. Fungsi pengisian ulang (isi ulang (C))

   Fungsi ini akan menginisialisasi ulang himpunan C, bila himpunan ini kosong.
6. Fungsi pembanding dan uji kelayakan.

   Fungsi ini akan melihat apakah kandidat yang diambil dari C apabila dimasukkan ke S akan membentuk *Point of Char* yang telah ada atau tidak. Jika belum maka masukkan ke elemen ke-i dari S, namun jika sudah masukkan ke elemen ke-i+1 dari S.

Penyelesaian persoalan ini menggunakan algoritma *greedy* akan dijelaskan lebih lanjut pada lampiran A.

2.2 Rancangan *Procedural*

Berikut merupakan rancangan prosedural algoritma *greedy* dalam bentuk *pseudocode*.

```
function greedy (input C: himpunan_kandidat)
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma
 greedy
Masukan: himpunan kandidat C
Keluaran: himpunan solusi yang bertipe himpunan_kandidat
}
```

```
Deklarasi
     x : kandidat
     S : himpunan_kandidat
Algoritma:
     S ← {} { inisialisasi S dengan kosong }
     while (not SOLUSI(S)) and (C ≠ {} ) do
           x ← SELEKSI(C) { pilih sebuah kandidat dari C}
           C ← C - {x} { elemen himpunan kandidat berkurang satu
     }
     if  LAYAK(S ∪{x}) then
           S ←  S ∪{x}
     endif
     endwhile
     {SOLUSI(S) or C = {} }
     if SOLUSI(S) then
           return S
     else
     write('tidak ada solusi')
     endif
```

2.3 Teori Kemungkinan *(Theory of Probability)*

Teori kemungkinan biasanya sering disebut dengan peluang. Peluang merupakan bagian dari ilmu matematika yang banyak dimanfaatkan dalam kehidupan sehari-hari. Kita sering menghadapi masalah dalam pengaturan suatu objek yang terdiri dari beberapa unsur baik yang disusun dengan mempertimbangkan urutan sesuai posisi yang diinginkan ataupun yang tidak. Misalnya menyusun kepanitiaan yang terdiri dari ketua, sekretaris dan bendahara

dimana urutan untuk posisi tersebut dipertimbangkan atau memilih beberapa orang untuk mewakili sekelompok orang dalam mengikuti suatu kegiatan yang dalam hal ini urutan tidak menjadi pertimbangan.

Dalam ilmu matematika, penyusunan objek yang terdiri dari beberapa unsur dengan mempertimbangkan urutan disebut dengan permutasi, sedangkan yang tidak mempertimbangkan urutan disebut dengan kombinasi. Berikut akan dijelaskan mengenai permutasi dan kombinasi.

2.3.1 Permutasi

Permutasi adalah teknik penyusunan objek dengan memperhatikan urutan objek tersebut. Secara formal permutasi dapat didefinisikan sebagai berikut:

Definisi 1

Permutasi dari n unsur yang berbeda $x_1, x_2,...,x_n$ *adalah pengurutan dari n unsur tersebut.*

Contoh:

Tentukan permutasi dari 3 huruf yang berbeda, misalnya ABC.

Permutasi dari huruf ABC adalah ABC, ACB, BAC, BCA, CAB, CBA.

Sehingga terdapat 6 permutasi dari huruf ABC.

Teorema 2.1

Terdapat n! permutasi dari n unsur yang berbeda.

Definisi 2.

Permutasi-r dari n unsur yang berbeda $x_1, x_2,\ldots, x_n$ *adalah pengurutan dari sub-himpunan dengan r anggota dari himpunan* $\{x_1, x_2,\ldots, x_n\}$*. Banyaknya permutasi-r dari n unsur yang berbeda dinotasikan dengan P(n, r).*

Contoh:

Tentukan permutasi-3 dari 5 huruf yang berbeda, misalnya ABCDE.

Permutasi-3 dari huruf ABCDE adalah

ABC ABD ABE ACB ACD ACE

ADB ADC ADE AEB AEC AED

BAC BAD BAE BCA BCD BCE

BDA BDC BDE BEA BEC BED

CAB CAD CAE CBA CBD CBE

CDA CDB CDE CEA CEB CED

DAB DAC DAE DBA DBC DBE

DCA DCB DCE DEA DEB DEC

EAB EAC EAD EBA EBC EBD

ECA ECB ECD EDA EDB EDC

Sehingga banyaknya permutasi-3 dari 5 huruf ABCDE adalah 60.

Teorema 2.2

Banyaknya permutasi-r dari n unsur yang berbeda adalah

$$P(n,r) = \frac{n!}{(n-r)!}$$

2.3.2 Kombinasi

Tidak seperti pada permutasi, kombinasi tidak mempertimbangkan urutan dalam penyusunan suatu objek.

Contoh:

Tentukan kombinasi-3 dari 5 huruf yang berbeda, misalnya ABCDE.

Kombinasi-3 dari huruf ABCDE adalah

ABC ABD ABE ACD ACE

ADE BCD BCE BDE CDE

Sehingga banyaknya kombinasi-3 dari 5 huruf ABCDE adalah 10.

Teorema 2.3

Banyaknya kombinasi-r dari n unsur yang berbeda adalah

$$C(n,r) = \frac{n!}{(n-r)!.r!}$$

2.4 Algoritma Greedy dengan Permutasi pada Kasus Penjadwalan Dokter Jaga

Seperti yang telah diuraikan pada bagian sebelumnya bahwa konsep algoritma *greedy* adalah menyelesaikan persoalan langkah per langkah *(step by step)* untuk menghasilkan solusi lokal yang akan mengarah ke solusi global. Alasan digunakan algoritma *greedy* dengan permutasi pada kasus ini adalah karena persoalan penyusunan jadwal dokter jaga di RSUD Arifin Achmad lebih mengarah pada sejumlah ketentuan *(constraint)* yang digunakan. Hal ini sesuai dengan salah satu konsep yang terdapat pada algoritma *greedy* yaitu, mengambil elemen dari himpunan kandidat (C) yang di uji kelayakannya berdasarkan sejumlah *constraint* yang ada, kemudian jika layak maka elemen kandidat yang diambil dapat dijadikan sebagai solusi.

Pada kasus penjadwalan dokter jaga yang dijadikan kandidat adalah dokter dan *constraint* yang berlaku sebagai uji kelayakan ditetapkan berdasarkan prosedur penyusunan jadwal jaga yang terdapat di RSUD Arifin Achmad

Pekanbaru. Algoritma *greedy* tidak selalu dapat menghasilkan solusi optimal, namun kelemahan tersebut dapat ditutupi dengan bantuan teori permutasi dimana teori ini berfungsi untuk membantu serta mempercepat penemuan solusi yaitu solusi penugasan dokter jaga.

2.5 RSUD Arifin Achmad Pekanbaru

RSUD Arifin Achmad merupakan Rumah Sakit yang telah berdiri di kota Pekanbaru provinsi Riau sejak tahun 1950-an. Rumah Sakit ini mengemban visi, menjadi Rumah Sakit pendidikan mandiri dengan pelayanan paripurna yang memenuhi standar internasional.

Untuk mencapai tujuannya, pihak Rumah Sakit ini selalu berupaya untuk selalu memberikan pelayanan yang terbaik bagi pasien. Hal ini juga berkaitan dengan dokter sebagai tenaga medis. Agar rutinitas antara dokter terhadap pasien dapat terkoordinir dengan baik, maka pihak Rumah Sakit melakukan penjadwalan terhadap dokter jaga. Dalam hal ini, penjadwalan untuk dokter jaga dibagi dalam tiga bagian instalasi, yaitu;

1. IRD (Instalasi Rawat Darurat)
2. IRNA (Instalasi Rawat Inap)
3. IRNA Utama (Instalasi Rawat Inap Utama)

Jam kerja dokter jaga dibagi atas tiga *shift*, yaitu:

P : Pagi 07.30 s/d 14.00

S : Sore 14.00 s/d 21.00

M : Malam 21.00 s/d 07.30

Penyusunan ulang terhadap jadwal dokter jaga dilakukan setiap bulan dan hasilnya dalam bentuk *hardfile* diserahkan kepada masing-masing dokter yang bertugas serta dilaporkan kepada Direktur Medik dan Keperawatan.

2.6 Teknik Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean (Pressman, 2002).

Teknik pengujian perangkat lunak dapat dibagi atas beberapa macam, namun yang paling umum digunakan adalah sebagai berikut:

1. Pengujian *white-box*

Pengujian *white-box* terkadang juga disebut sebagai pengujian *glass-box*, merupakan metode perancangan *test case* yang menggunakan struktur kontrol dari perancangan prosedural untuk memperoleh *test case*.

Dengan menggunakan metode *white-box*, sistem analis akan memperoleh *test case* yang:

a. Memberi jaminan bahwa jalur *independent* pada suatu modul telah digunakan paling tidak satu kali.
b. Menggunakan semua keputusan logis pada sisi *true* dan *false*.
c. Mengerjakan seluruh *loop* yang sesuai dengan batasannya.
d. Menggunakan struktur data internal yang menjamin validitas.

2. Pengujian *black-box*

Pengujian *black-box* adalah pengujian yang fokus pada persyaratan fungsional perangkat lunak. Pengujian ini memungkinkan sistem analis

memperoleh kumpulan kondisi input yang akan mengerjakan seluruh keperluan fungsional program.

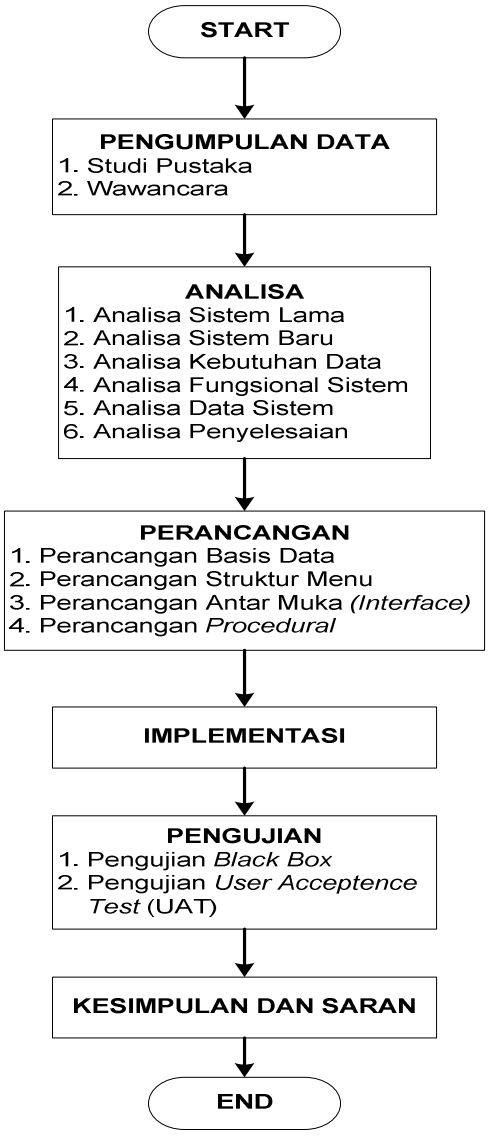
Tujuan metode ini mencari kesalahan pada:

a. Fungsi yang tidak benar atau hilang
b. Kesalahan pada *interface*
c. Kesalahan pada struktur data atau akses *database*
d. Kesalahan kinerja
e. Kesalahan inisialisasi dan tujuan akhir

BAB III

METODOLOGI PENELITIAN

Metodologi penelitian merupakan sistematika tahapan yang dilaksanakan selama pembuatan tugas akhir. Secara garis besar metodologi penelitian tugas akhir ini dapat dilihat pada gambar 3.1.



Gambar 3.1 *Flowchart* Metodologi Penelitian

Berdasarkan *flowchart* pada gambar 3.1 metodologi penelitian dalam pengerjaan tugas akhir meliputi lima tahapan, yaitu:

3.1 Proses Pengumpulan Data

Pengumpulan data merupakan tahap awal dari penelitian, dalam hal ini pengumpulan data dapat meliputi:

1. Studi Pustaka

   Studi pustaka dalam penelitian tugas akhir ini dapat dilakukan dengan mempelajari beberapa *literature*, misalnya melalui media internet maupun buku-buku yang berhubungan dengan penelitian yang meliputi; konsep algoritma *greedy*, teori kemungkinan permutasi serta pengaplikasian *greedy* pada kasus penjadwalan.

2. Wawancara

   Melakukan wawancara langsung dengan pihak terkait di RSUD Arifin Achmad, untuk memperoleh data yang akurat dan mengetahui prosedur-prosedur serta kondisi penjadwalan yang sebenarnya.

3.2 Analisa

Analisa permasalahan berkaitan dengan mengidentifikasi kebutuhan dalam suatu penelitian. Analisa dapat terbagi lagi atas beberapa tahapan, antara lain sebagai berikut:

3.2.1 Analisa Sistem Lama

Pada tahap ini dilakukan analisa terhadap sistem lama atau metode pengerjaan yang sedang berlangsung, termasuk untuk mengetahui kelemahan yang dimiliki oleh sistem lama tersebut.

3.2.2 Analisa Sistem Baru

Setelah menganalisa sistem lama, maka tahapan dapat dilanjutkan dengan menganalisa sistem yang baru. Dalam tahapan ini, akan diidentifikasi cara kerja dari sistem baru yang akan dibangun.

3.2.3 Analisa Kebutuhan Data

Tahapan ini dilakukan untuk mengidentifikasi variabel. Variabel merupakan objek penelitian atau sesuatu hal yang menjadi titik perhatian dalam suatu penelitian. Variabel adalah data yang dibutuhkan dalam pembuatan sistem. Untuk itu menganalisa atau mengidentifikasi variabel merupakan syarat mutlak penelitian. Semakin dalam pengidentifikasian variabel, maka data yang diperoleh akan semakin luas sehingga gambaran hasil penelitian menjadi semakin teliti.

3.2.4 Analisa Fungsional Sistem

Analisa yang digunakan pada sistem adalah dengan pemodelan fungsional. Pemodelan fungsional merupakan pemodelan yang menggambarkan suatu masukkan yang diproses pada sistem menjadi keluaran yang dibutuhkan bagi pengguna sistem. Pada tahapan ini, akan dibahas mengenai *Flowchart* serta *Data Flow Diagram,* yang terdiri dari *Context Diagram* level 0, DFD Level 1 sampai DFD level 3.

3.2.5 Analisa Data Sistem

Pada tahapan ini, data sistem akan dirancang menggunakan *Entity Relationship Diagram* (ERD).

3.2.6 Analisa Penyelesaian

Pada tahapan ini dilakukan analisa penyelesaian terhadap kasus permasalahan, dalam hal ini menggunakan algoritma *greedy* dengan permutasi.

3.3 Perancangan

Setelah melakukan analisa, maka kemudian dilanjutkan dengan perancangan sistem berdasarkan analisa permasalahan yang telah dilakukan sebelumnya.

3.3.1 Perancangan Basis Data

Setelah menganalisa sistem yang akan dibuat, maka tahap selanjutnya adalah analisa dan perancangan basis data yang dilakukan untuk melengkapi komponen sistem.

3.3.2 Perancangan Struktur Menu

Rancangan struktur menu diperlukan untuk memberikan gambaran terhadap menu-menu atau *fitur* pada sistem yang akan dibangun.

3.3.3 Perancangan Antar Muka (*Interface)*

Untuk mempermudah komunikasi antara sistem dengan pengguna, maka perlu dirancang antar muka *(interface)*. Dalam perancangan *interface* hal terpenting yang ditekankan adalah bagaimana menciptakan tampilan yang baik dan mudah dimengerti oleh pengguna.

3.3.4 Perancangan *Procedural*

Perancangan *procedural* merupakan tahap perancangan pada metode atau algoritma yang akan digunakan dalam membangun sistem.

3.4 Implementasi

Setelah analisa dan perancangan sistem selesai, maka tahap selanjutnya adalah implementasi. Implementasi adalah tahapan dimana dilakukan *coding* atau pengkodean dan sistem telah siap untuk dioperasikan pada keadaan yang sebenarnya. Untuk implementasi sistem akan dilakukan pada komputer pembuat sistem dengan spesifikasi sebagai berikut :

Operating System	: *Windows XP Professional*
Processor	: Intel Core 2 Duo 2.00 GHz
RAM	: 1 GB
Harddisk	: 160 GB
Bahasa Pemrograman	: *Ms. Visual Basic 6.0*
Database	: *Ms.Access 2003*

3.5 Pengujian

Pengujian merupakan tahapan dimana aplikasi akan dijalankan, tahap ini diperlukan untuk mengetahui apakah sistem sesuai dengan tujuan yang ingin dicapai. Pada tahapan ini akan dilakukan pengujian terhadap perangkat lunak menggunakan metode pengujian sebagai berikut:

3.5.1 Pengujian *black box*

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Pengujian ini memungkinkan sistem analis untuk memperoleh kondisi *input* yang akan mengerjakan seluruh keperluan fungsional sistem.

3.5.2 Pengujian *User Acceptence Test* (UAT)

Pengujian *User Acceptence Test* merupakan jenis pengujian dengan menggunakan kuesioner atau angket yang berisi pertanyaan-pertanyaan seputar tugas akhir.

3.6 Kesimpulan dan Saran

Kesimpulan dan saran merupakan tahapan akhir dari sebuah penelitian. Kesimpulan dapat bernilai positif maupun negatif, hal ini sesuai dengan hasil yang diperoleh pada pengujian sistem. Sedangkan saran adalah harapan untuk masa yang akan datang bagi perkembangan sistem selanjutnya.

BAB IV

ANALISA DAN PERANCANGAN

Analisa merupakan tahapan yang penting dalam pembuatan sistem berbasis komputer. Analisa merupakan proses pengidentifikasian kebutuhan dalam pembuatan sistem. Setelah proses atau tahapan analisa selesai, maka dilanjutkan dengan tahap perancangan berdasarkan analisa permasalahan yang telah dilakukan. Perancangan yang dibuat harus memiliki kesesuaian dengan analisa sistem yang telah dilakukan pada tahapan sebelumnya.

4.1 Analisa

Analisa yang akan diuraikan pada bab ini terdiri dari beberapa tahapan, antara lain sebagai berikut;

4.1.1 Analisa Sistem Lama

RSUD Arifin Achmad sebelumnya masih melakukan penjadwalan dokter jaga secara manual. Penjadwalan dibagi atas tiga bagian instalasi, yaitu; IRNA (Instalasi Rawat Inap), IRNA Utama (Instalasi Rawat Inap Utama) dan IRD (Instalasi Rawat Darurat). Pihak yang berwenang terhadap penjadwalan pada masing-masing instalasi menerima daftar nama dokter yang bertugas serta menerima *request* jadwal secara langsung dari dokter yang bersangkutan, hal ini berlaku untuk IGD dan IRNA Utama. Sedangkan pada IRNA Biasa jadwal disusun berdasarkan kebijakan kepala instalasi, sehingga para dokter harus

mengikuti jadwal jaga yang telah ditetapkan. Penjadwalan ulang dilakukan setiap awal bulan. Jadwal yang telah disusun ditulis kembali dengan menggunakan lembar kerja berbentuk kertas *(excel)*, kemudian diserahkan kepada masing-masing dokter yang bertugas serta dilaporkan kepada Komite Medik/ Direktur Medik dan Keperawatan.

Beberapa hal yang menjadi bahan pertimbangan dalam penyusunan jadwal dokter jaga adalah sebagai berikut:

1. Pada bagian IRNA (Instalasi Rawat Inap), Penjadwalan dokter jaga dilakukan sesuai dengan kebijakan kepala instalasi, sehingga dokter-dokter yang bertugas harus mengikuti jadwal yang telah disusun.
2. Pada bagian IRNA Utama (Instalasi Rawat Inap Utama), Dokter-dokter yang bertugas dapat melakukan *request* atau menyesuaikan waktu jaga mereka dengan jadwal praktek di Rumah Sakit lain.
3. Pada bagian IRD (Instalasi Rawat darurat), penjadwalan dilakukan oleh pihak yang telah diberikan wewenang dan dokter yang bertugas pada bagian ini, dapat menyesuaikan waktu jaga dengan jadwal praktek mereka di tempat lain.
4. Satu shift jaga terdiri dari satu orang dokter, hal ini berlaku untuk IRNA dan IRNA Utama. Sedangkan pada IRD, satu shift terdiri dari dua orang dokter jaga.
5. Untuk IRNA, *shift* pagi berlaku pada hari libur saja.
6. Dokter dari satu instalasi tidak dapat pindah jaga ke instalasi lainnya.

7. Dokter yang memperoleh *shift* malam, diperbolehkan untuk libur pada keesokan hari.
8. Penjadwalan ulang dilakukan setiap awal bulan, oleh sebab itu harus diperoleh laporan mengenai dokter yang akan mengambil cuti sehingga tidak terjadi kendala pada saat jadwal telah disusun.
9. Apabila dokter yang bersangkutan berhalangan hadir sesuai dengan jadwalnya, maka diharapkan dokter tersebut untuk segera mencari pengganti.

Jam kerja dokter jaga dibagi menjadi tiga *shift*, yaitu;

P : Pagi 07.30 s/d 14.00

S : Sore 14.00 s/d 21.00

M : Malam 21.00 s/d 07.30

Permasalahan yang dihadapi dalam penyusunan jadwal ini adalah membutuhkan waktu yang relatif lebih lama, hal ini disebabkan:

1. Ketika melakukan penjadwalan terlebih dahulu harus diperiksa satu persatu *shift* dan tanggal yang masih kosong serta menyesuaikannya dengan jadwal *(schedule)* dokter yang bersangkutan. Proses ini akan menghabiskan waktu terutama bila jadwal yang disusun cukup banyak.
2. Apabila pihak yang bertanggung jawab dalam penjadwalan diganti, maka pihak yang baru ditunjuk harus mempelajari dari awal seluruh ketentuan atau aturan yang digunakan dalam penyusunan jadwal tersebut.

4.1.2 Analisa Sistem Baru

Sistem baru dibangun berdasarkan aturan atau ketentuan yang menjadi bahan pertimbangan pada sistem penjadwalan dokter jaga yang lama, hal ini disebut sebagai batasan *(constraints)*. Oleh sebab itu, penjadwalan dokter jaga dapat dibagi atas tiga tahapan, yaitu;

a. Penentuan *Constraints*

   Penentuan *constraint* merupakan tahap untuk menentukan seluruh kriteria atau aturan yang digunakan pada penyusunan jadwal dokter jaga.

b. Penghitungan *Shift (Enumeration Shift)*

   Penghitungan *shift* adalah proses penghitungan total *shift* jaga yang diperlukan selama periode 1 bulan, sehingga perlu dilakukan pengecekan jumlah hari dari bulan yang dijadwalkan. Penghitungan *shift* diperlukan untuk menghasilkan maksimum perolehan *shift* jaga oleh masing-masing dokter dalam 1 bulan.

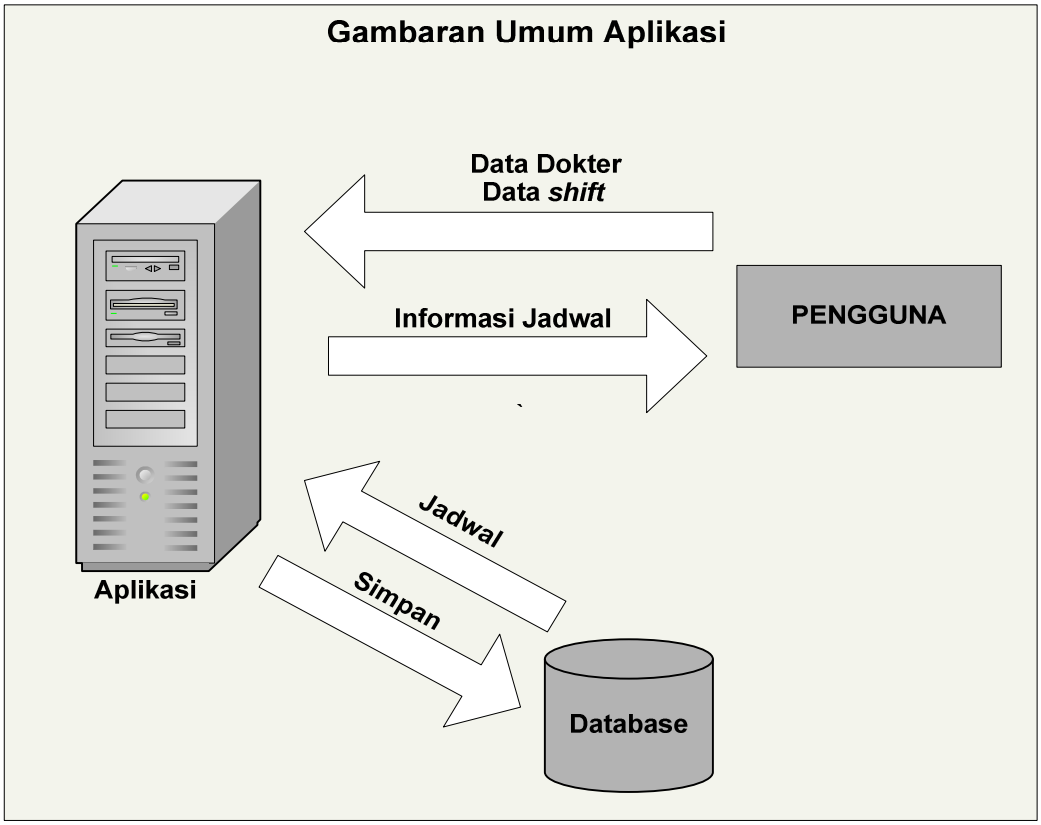
c. Penugasan Dokter *(Doctor Assignation)*

   Penugasan dokter adalah tahap penyusunan jadwal jaga untuk menemukan penugasan yang layak dengan mempertimbangkan batasan-batasan *(constraints)* yang ada. Pada tahap ini dilakukan proses pengacakan atau *random* terhadap dokter pada masing-masing instalasi, dengan menggunakan algoritma *greedy* permutasi. Proses *random* diasumsikan dengan membentuk kelompok-kelompok dokter, dimana posisi dokter dalam kelompok ikut dipertimbangkan. Kelompok yang dihasilkan dari *random*

dokter ini digunakan untuk menemukan penugasan jaga yang layak bagi masing-masing dokter.

Sistem ini menyediakan tabel khusus yang berisi *request* jadwal jaga serta *day off* dokter. Jadwal yang telah di *request* oleh dokter secara otomatis mengisi tabel jadwal jaga, kemudian masing-masing dokter yang tidak sedang *day off* dipertimbangkan untuk mengisi jadwal *(shift)* yang masih kosong.

Sistem yang dibuat adalah sistem berbasis *windows* yang mengelola penjadwalan dokter jaga. Sistem ini hanya digunakan oleh satu pengguna saja, yaitu pihak yang telah diserahkan wewenang untuk melakukan penjadwalan pada masing-masing bagian atau instalasi. Gambaran umum aplikasi yang akan dibangun dapat dilihat melalui gambar 4.1.



Gambar 4.1 Gambaran Umum Aplikasi

Gambar 4.1 menjelaskan bahwa pengguna menginputkan data dokter beserta data *shift* sebagai data master. Kemudian sistem akan mengolah data tersebut sesuai dengan algoritma yang diterapkan dan hasilnya berupa jadwal ditempatkan dalam suatu *database*. Pengguna dapat memperoleh informasi dari sistem berupa jadwal yang diinginkan.

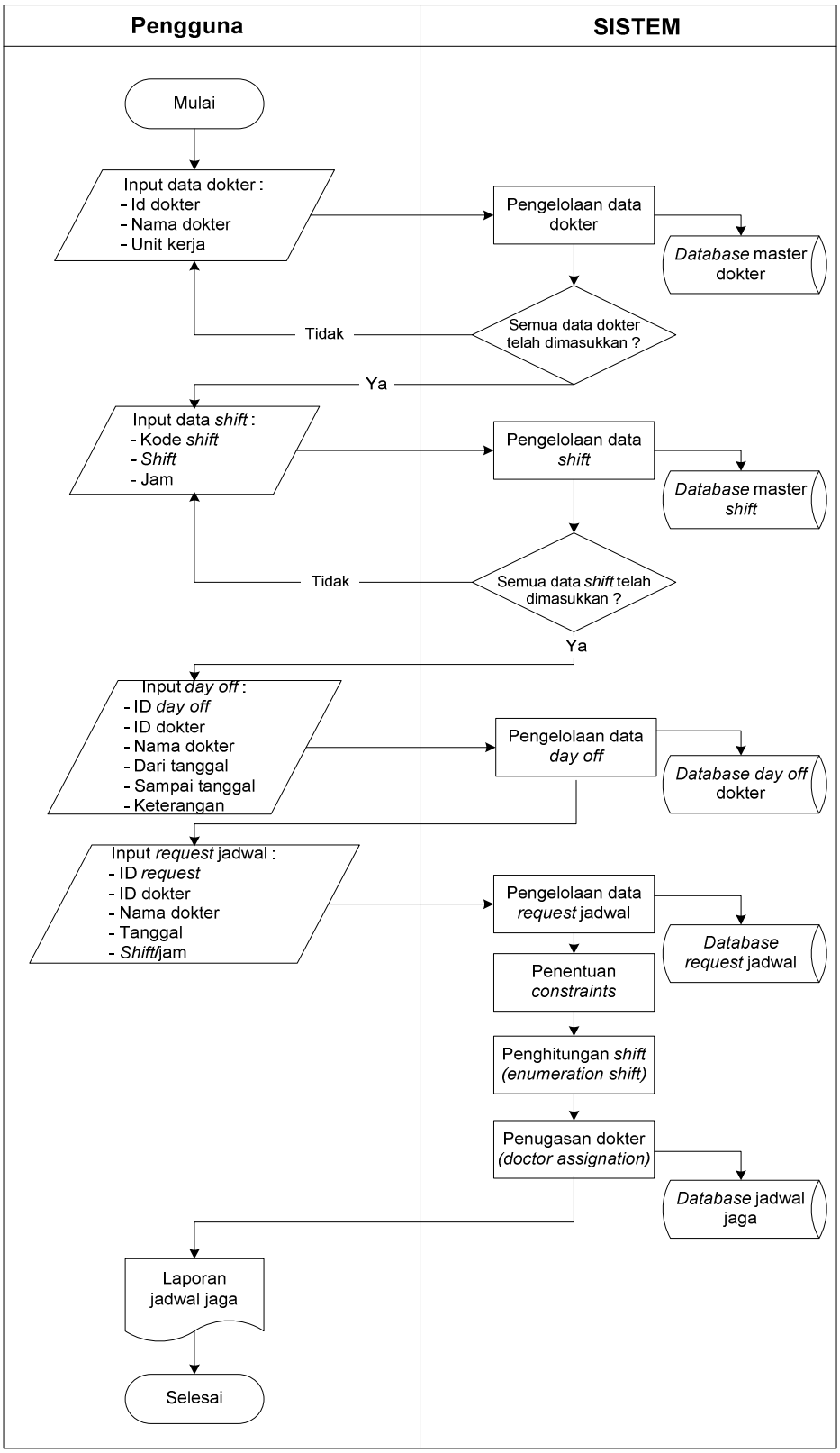
4.1.3 Analisa Kebutuhan Data

Penjadwalan dokter jaga membutuhkan data-data yang meliputi:

1. Data dokter, terdiri dari; Id dokter, nama dokter, serta unit kerja (nama instalasi penugasan dokter).
2. Data *shift*, terdiri dari; kode *shift*, *shift* serta jam jaga.
3. Data *day off*, terdiri dari; Id *day off*, Id dokter, tanggal mulai *off*, tanggal akhir *off*, dan keterangan.
4. Data *request* jadwal, terdiri dari; Id *request*, Id dokter, tanggal serta *shift*/jam jaga yang di *request*.

4.1.4 *Flowchart* Sistem Penjadwalan Dokter Jaga

Flowchart untuk sistem penjadwalan dokter jaga dapat dilihat pada gambar 4.2.



Gambar 4.2 *Flowchart* Sistem Penjadwalan Dokter Jaga

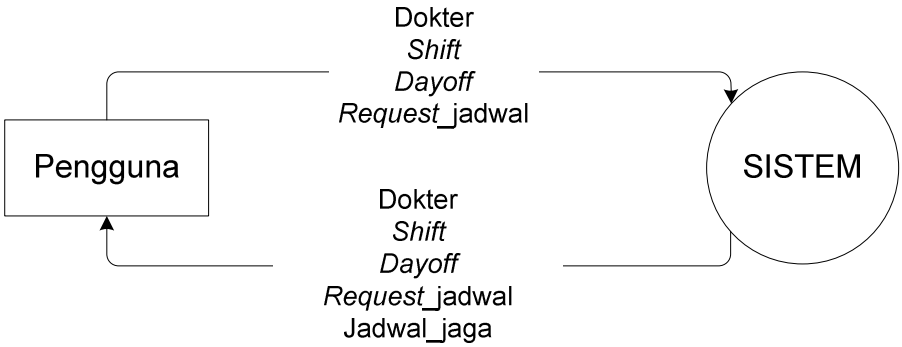
4.1.5 Analisa Model Fungsional Sistem

Pemodelan fungsional merupakan pemodelan yang menggambarkan suatu masukkan yang diproses pada sistem, menjadi keluaran yang dibutuhkan bagi pengguna sistem.

Pemodelan fungsional sistem ini meliputi *Context Diagram* (Diagram konteks) dan *Data Flow Diagram* (DFD). Analisa fungsional sistem lebih lanjut dapat dilihat pada lampiran B.

4.1.5.1 *Context Diagram*

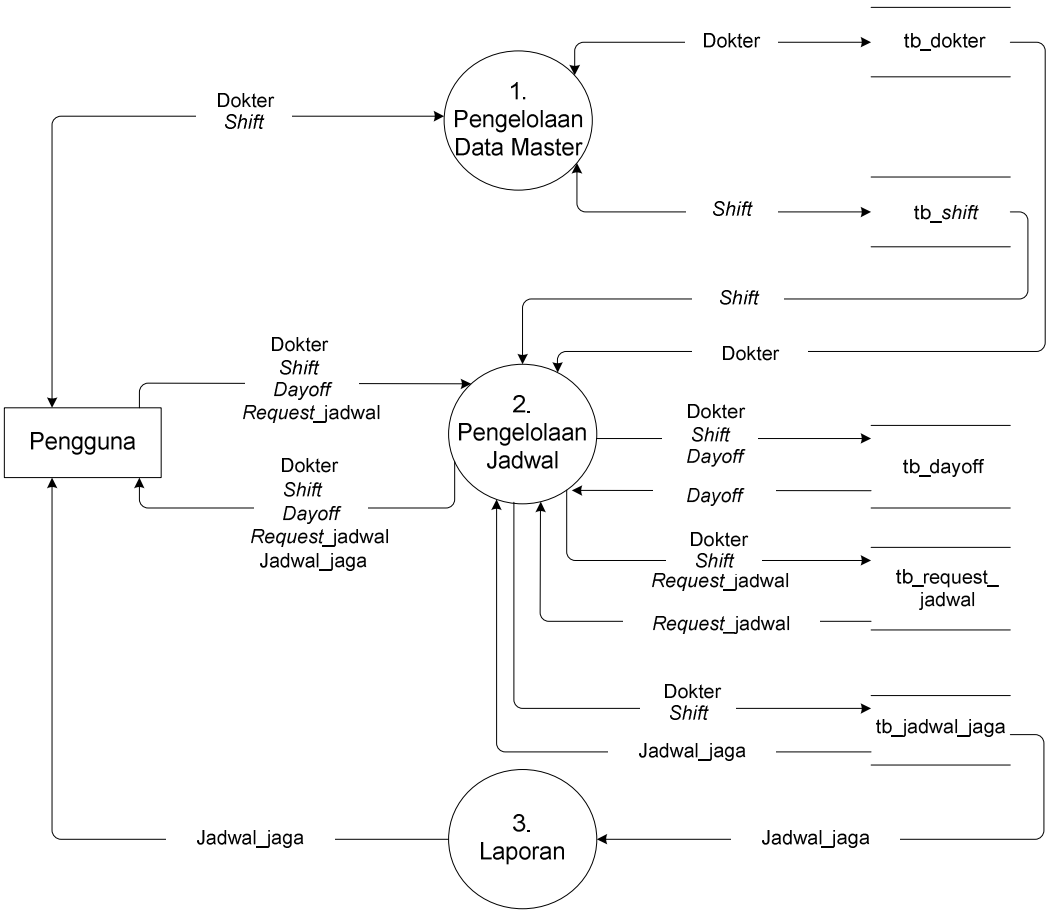
Context Diagram yang menggambarkan proses kerja sistem secara umum, dapat dilihat pada gambar 4.3.



Gambar 4.3 *Context Diagram*

4.1.5.2 DFD Level 1

DFD level 1 dapat dilihat dari gambar 4.4.



Gambar 4.4 DFD Level 1

Tabel 4.1 Proses DFD Level 1

Nama	Deskripsi
Pengelolaan Data Master	Berisi proses pengelolaan data master
Pengelolaan Jadwal	Berisi proses pengelolaan jadwal
Laporan	Berisi proses pelaporan

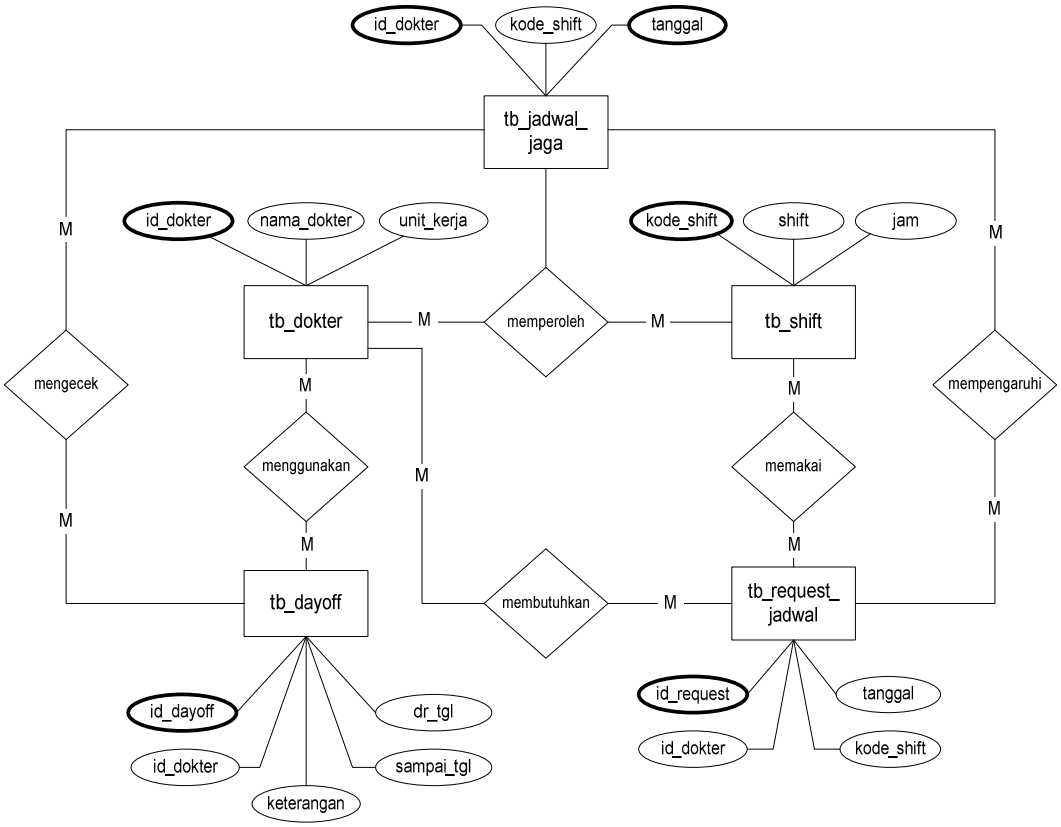
Tabel 4.2 Aliran Data DFD Level 1

Nama	Deskripsi
Dokter	Data dokter jaga yang bertugas
Shift	Data *shift* jaga

Dayoff	Data *day off* dokter
*Request*_jadwal	Data *request* jadwal jaga
Jadwal_jaga	Informasi jadwal jaga

4.1.6 Analisa Data Sistem

Analisa data menggambarkan data-data yang terlibat dalam proses pada sistem atau disebut dengan *database*. Analisa data diperlihatkan pada ERD *(Entity Relationship Diagram)* seperti pada gambar 4.5.



Gambar 4.5 *Entity Relationship Diagram* (ERD)

Tabel 4.3 Keterangan Entitas pada ERD

No	Nama	Deskripsi	Atribut	Primary key	Foreign Key
1.	tb_dokter	Menyimpan data master dokter	- id_dokter - nama_dokter - unit_kerja	- id_dokter	-
2.	tb_shift	Menyimpan data master *shift*.	- kode_shift - shift - jam	- kode_shift	-
3.	tb_dayoff	Menyimpan data jadwal *off* dokter.	- id_dayoff - id_dokter - dr_tgl - sampai_tgl - keterangan	- id_dayoff	- id_dokter
4.	tb_request_ Jadwal	Menyimpan data *request* jadwal jaga	- id_request - id_dokter - tanggal - kode_shift	- id_request	- id_dokter - kode_shift
5.	tb_jadwal_ jaga	Menyimpan data jadwal dokter jaga.	- id_dokter - tanggal - kode_shift	- id_dokter - tanggal	- id_dokter - kode_shift

Tabel 4.4 Hubungan pada ERD

No.	Nama	Deskripsi
1.	Memperoleh	Hubungan antara entitas tb_dokter, tb_shift dan entitas tb_jadwal_jaga.
2.	Mempengaruhi	Hubungan antar entitas tb_request_jadwal dengan entitas tb_jadwal_jaga.

3.	Mengecek	Hubungan antara entitas tb_jadwal_jaga dengan entitas tb_dayoff.
4.	Menggunakan	Hubungan antara entitas tb_dokter dengan entitas tb_dayoff
5.	Membutuhkan	Hubungan antara entitas tb_dokter dengan entitas tb_request_jadwal.
5.	Memakai	Hubungan antara entitas tb_shift dengan entitas tb_request_jadwal.

4.1.7 Analisa Penyelesaian

Penjadwalan dokter jaga dibagi atas tiga instalasi, yaitu; IRNA (Instalasi Rawat Inap), IRNA Utama (Instalasi Rawat Inap Utama), dan IRD (Instalasi Rawat Darurat). Dimana dalam hal ini tidak berlaku rotasi atau pindah jaga terhadap dokter dari satu instalasi ke instalasi lain dan jadwal disusun untuk periode 1 bulan.

Sistem penjadwalan dokter jaga dibagi atas tiga tahapan, yaitu;

4.1.7.1 Penentuan *Constraints*

Masing-masing instalasi memiliki ketentuan *(constraints)* yang digunakan pada penyusunan jadwal jaga. Sistem penjadwalan dokter jaga pada IRNA (Instalasi Rawat Inap) memiliki ketentuan *(constraints)* sebagai berikut:

1. Dokter tidak dapat melakukan *request* jadwal jaga.
2. Tiap *shift* terdiri atas satu orang dokter jaga.

3. Sistem jaga pagi (P), sore (S), dan malam (M) hanya berlaku untuk hari minggu. Sedangkan pada hari kerja (non-minggu) berlaku sistem jaga sore (S) dan malam (M).
4. Masing-masing dokter hanya memperoleh 1 *shift* dalam sehari.
5. Dokter yang memperoleh *shift* malam (M) keesokan hari diberi libur.
6. Hari libur nasional dianggap sebagai hari kerja, sehingga berlaku *shift* sore (S) dan malam (M).

Constraint penjadwalan dokter jaga pada IRNA Utama (Instalasi Rawat Inap Utama) adalah sebagai berikut:

1. Dokter dapat melakukan *request* jadwal jaga.
2. Tiap *shift* terdiri atas satu orang dokter.
3. Tiap hari berlaku sistem jaga pagi (P), sore (S), dan malam (M).
4. Masing-masing dokter hanya memperoleh 1 *shift* dalam sehari.
5. Dokter yang memperoleh *shift* malam (M) keesokan hari diberi libur.

Penjadwalan dokter jaga pada IRD (Instalasi Rawat Darurat) memiliki *constraint* sebagai berikut:

1. Dokter dapat melakukan *request* jadwal jaga.
2. Tiap *shift* terdiri atas dua orang dokter.
3. Tiap hari berlaku *shift* pagi (P), sore (S), dan malam (M).
4. Masing-masing dokter hanya memperoleh 1 *shift* dalam sehari.
5. Dokter yang memperoleh *shift* malam (M) keesokan hari diberi libur.

4.1.7.2 Penghitungan *Shift (Enumeration Shift)*

Pada tahap ini dilakukan pengecekan jumlah hari dalam 1 bulan serta jumlah hari minggu dan non-minggu jika diperlukan. Total *shift* yang dibutuhkan dalam periode 1 bulan akan dibagi dengan jumlah dokter yang ada pada tiap instalasi, sehingga akan menghasilkan nilai maksimum perolehan jaga masing-masing dokter. Penghitungan tersebut dapat dinyatakan dengan rumus:

$$\text{Total } shift = 3 \times \text{Jumlah hari} \quad \text{..... (4.1)}$$

atau

$$\text{Total } shift = 2 \times \text{Jumlah hari} \quad \text{..... (4.2)}$$

Keterangan Rumus:

Total *shift* = jumlah seluruh *shift* yang dibutuhkan selama 1 bulan

Jumlah hari = jumlah hari

Penggunaan kedua rumus diatas disesuaikan dengan kebutuhan masing-masing instalasi. Rumus (4.1) digunakan untuk menghitung total *shift* pada hari yang berlaku sistem jaga pagi, sore dan malam (berlaku 3 *shift*). Sedangkan Rumus (4.2) digunakan untuk menghitung total *shift* pada hari yang berlaku sistem jaga sore dan malam (berlaku 2 *shift*).

$$\text{Maksimum jaga} = \frac{\text{Total } shift}{\text{Jumlah dokter}} \quad \text{.....(4.3)}$$

Keterangan Rumus:

Maksimum jaga = nilai maksimum jaga yang diperoleh masing-masing dokter dalam periode 1 bulan

Total *shift* = jumlah seluruh *shift* yang dibutuhkan selama 1 bulan

Jumlah dokter = jumlah dokter dalam satu instalasi

4.1.7.3 Penugasan Dokter *(Doctor Assignation)*

Penugasan dokter merupakan tahapan untuk menemukan penugasan yang layak bagi dokter jaga. Untuk itu dilakukan pengacakan *(random)* dokter tiap instalasi, sehingga proses *random* dapat diasumsikan sebagai proses pengelompokan dokter. Proses ini menggunakan algoritma *greedy* permutasi, dimana permutasi digunakan untuk mengarahkan algoritma *greedy* yang dibuat sehingga proses *random* memperhatikan posisi dokter dalam kelompok yang dihasilkan. Pada penugasan dokter dilakukan penelusuran terhadap kelompok-kelompok tersebut, untuk mencari dokter yang layak ditugaskan pada *shift* jaga yang masih kosong.

Proses *random* dokter pada kasus ini dibagi atas dua macam, yaitu;

a). *Random* untuk dua dokter.

Proses *random* yang dilakukan untuk menghasilkan kelompok, yang mana tiap kelompok terdiri dari dua orang dokter. Proses *random* ini digunakan untuk hari yang berlaku sistem jaga sore dan malam (berlaku 2 *shift*). Kemungkinan solusi kelompok yang dihasilkan dapat dihitung dengan merujuk pada teorema 2.2, yaitu dengan menggunakan permutasi 2 dengan rumus:

$$\text{Solusi kelompok} = \frac{n!}{(n-2)!} \quad \textbf{(4.4)}$$

n = jumlah dokter dalam satu instalasi.

b). *Random* untuk tiga dokter

Proses *random* yang digunakan untuk menghasilkan kelompok, yang mana tiap kelompok terdiri dari tiga orang dokter. Proses ini digunakan untuk hari yang menggunakan sistem jaga pagi, sore dan malam (berlaku 3 *shift*). Kemungkinan jumlah kelompok yang dihasilkan dapat dihitung dengan merujuk pada teorema 2.2, yaitu menggunakan permutasi 3 dengan rumus:

$$\text{Solusi kelompok} = \frac{n!}{(n-3)!} \quad \text{..... } \mathbf{(4.5)}$$

n = jumlah dokter dalam satu instalasi

Struktur data dan komponen *greedy* yang dibutuhkan dalam penugasan dokter jaga adalah sebagai berikut:

1. *Array of Integer* (C)

   Himpunan seluruh dokter pada masing-masing instalasi dan merupakan himpunan simbol kandidat dokter yang diambil berdasarkan *id Array of Integer* yang dibentuk.

2. *Array of Array of Array Integer* (*Integer* x, *Integer* y, *Integer* z)

   Untuk memasukkan dokter-dokter yang terpilih dalam proses acak *(random)* dan merupakan solusi (S).

3. Fungsi pemilihan (pilih (C))

   Fungsi untuk memilih dokter dari kandidat dokter (C) sesuai dengan kriteria algoritma yang diterapkan. Pada kasus ini pemilihan dilakukan dari dokter dengan *id Array* terkecil.

4. Fungsi pengisian ulang (isi ulang (C))

   Fungsi ini diperlukan untuk menginisialisasi ulang (C), jika himpunan ini kosong.

5. Fungsi kelayakan

   Fungsi ini untuk menguji kelayakan dokter yang dipilih dari kandidat dokter (C) berdasarkan kriteria *(constraints)* yang telah ditentukan.

Berdasarkan komponen-komponen diatas langkah-langkah *random* dokter dapat diuraikan sebagai berikut:

1. Inisialisasi solusi (S) yang kosong.

   Kemungkinan Indeks solusi yang dapat diperoleh dihitung menggunakan rumus permutasi, yaitu; Rumus (4.4) untuk *random* dua dokter atau Rumus (4.5) untuk *random* tiga dokter.

2. Pilih dokter dari kandidat dokter (C) dengan menggunakan fungsi pemilihan.
3. Masukkan dokter yang terpilih pada *Array of Integer* (jika x berisi maka tempatkan dokter pada posisi y, jika y berisi maka tempatkan pada posisi z).
4. Uji kelayakan, apakah dokter yang diambil jika dimasukkan ke S akan menghasilkan kelompok yang telah ada atau tidak. Jika belum, maka tempatkan sebagai solusi dan hapus dokter dari (C). Pada proses inilah berlaku permutasi, dimana posisi dokter dalam kelompoknya turut diperhitungkan.
5. Setelah C kosong dan S belum penuh, maka isi ulang C dan ulangi kembali dari tahap 2. Jika seluruh kelompok telah terbentuk (sesuai dengan jumlah indeks solusi), maka hapus indeks solusi yang tidak penuh.

6. Lakukan inisialisasi terhadap dokter dalam kelompoknya. Ketentuan inisialisasi dokter dalam kelompok adalah sebagai berikut:
    a) Untuk *random* yang menghasilkan dua dokter, elemen pada (*Integer* x, *Integer* y) diinisialisasi sebagai (S,M), artinya dokter yang mengisi *Array of Integer* x akan memperoleh *shift* sore (S) dan dokter pada *Array of Integer* y akan memperoleh *shift* malam (M).
    b) Untuk *random* yang menghasilkan tiga dokter, elemen pada (*Integer* x, *Integer* y, *Integer* z) diinisialisasikan sebagai (P,S,M). Artinya dokter pada *Array of Integer* x diberi *shift* pagi (P), *Array of Integer* y diberi *shift* sore (S) dan dokter pada *Array of Integer* z diberi *shift* malam (M).

Setelah tahap penentuan *constraint* dan penghitungan *shift*, maka tahap penugasan dokter jaga adalah sebagai berikut:

1. Cek tanggal penugasan untuk mengetahui *shift* yang masih kosong, kemudian proses *greedy* dilakukan untuk mengisi *shift* tersebut.
2. Lakukan parsing terhadap kelompok yang ditelusuri dari (S) ke (C). Artinya solusi berupa kelompok yang telah diperoleh akan menjadi kandidat baru yaitu kandidat kelompok.
3. Inisialisasi S yang kosong.
4. Pilih dokter dari kandidat kelompok (C). Proses penelusuran dimulai sesuai urutan kelompok.
5. Uji kelayakan dokter dalam kelompok, apakah sesuai untuk ditempatkan pada *shift* yang kosong tersebut atau tidak. Kriteria atau *Constraint* yang digunakan untuk uji kelayakan, antara lain; dokter yang bersangkutan

tidak sedang *day off*, tidak melewati nilai maksimum perolehan *shift* jaga dan dokter yang memperoleh *shift* malam untuk keesokan harinya tidak diperhitungkan.

6. Jika layak, masukkan dokter sebagai solusi (S), dokter yang tidak layak dihapus.
7. Setelah C kosong dan seluruh solusi belum tercapai, dalam arti masih terdapat shift yang kosong, maka isi ulang C dan ulangi tahap 4.

Untuk lebih jelas berikut ini diuraikan analisa penyelesaian penjadwalan dokter jaga pada masing-masing instalasi.

4.1.7.4 Penyelesaian Penjadwalan Dokter Jaga IRNA (Instalasi Rawat Inap)

Analisa penyelesaian penjadwalan dokter jaga pada IRNA tahap demi tahap adalah sebagai berikut:

1. Penghitungan *shift (enumeration shift)*

Langkah-langkah dalam penghitungan *shift* jaga adalah sebagai berikut:

1. Cek jumlah hari dalam 1 bulan, misal bulan januari 2010 terdapat 31 hari.
2. Pada instalasi ini terdapat perbedaan sistem jaga pada hari minggu dan non-minggu, maka cek jumlah hari minggu dan non-minggu.

   Hari minggu = 5 hari Non-minggu = 26 hari
3. Hitung jumlah *shift* yang dibutuhkan selama 1 bulan.

   Jumlah *shift* untuk hari minggu = 3 x 5 hari = 15 *shift* (berlaku *shift* pagi, sore, malam).

Jumlah *shift* untuk hari non-minggu = 2 x 26 hari = 52 *shift* (berlaku *shift* sore, malam).

Total *shift* = 15 + 52 = 67 *shift*

4. Hitung nilai maksimum jaga yang dapat diperoleh masing-masing dokter dengan cara membagi total *shift* dengan jumlah dokter.

   Misalnya instalasi ini terdiri dari 4 orang dokter, maka;

   Maksimum jaga = 67/4 = 17 *shift*

   Artinya masing-masing dokter dapat memperoleh maksimal 17 *shift* jaga dalam 1 bulan.

2. Penugasan dokter *(doctor assignation)*

Instalasi ini memiliki sistem jaga yang berbeda, yaitu pada hari minggu berlaku tiga *shift* dan pada non-minggu berlaku dua *shift*. Sehingga dibutuhkan proses *random* untuk dua dokter dan tiga dokter.

Terdapat 4 orang dokter, C = {1,2,3,4}.

Sehingga kelompok-kelompok yang dihasilkan dari kedua proses *random* dan merupakan solusi dari *greedy* permutasi adalah sebagai berikut.

(S) = {(1,2) (3,4) (1,3) (2,4) (1,4) (2,3) (2,1) (3,2) (4,3) (4,1) (3,1) (4,2)}

(S) = {(1,2,3) (4,1,2) (3,4,1) (2,3,4) (1,2,4) (3,1,2) (3,4,2) (1,3,4) (1,3,2) (2,4,1) (3,1,4) (4,2,3) (1,4,2) (2,3,1) (3,2,4) (4,1,3) (1,4,3) (2,1,4) (3,2,1) (2,4,3) (2,1,3) (4,2,1) (4,3,1) (4,3,2)}

Keterangan : Proses *random* dilakukan pada simbol angka yang merupakan *id Array of Integer* untuk mewakili masing-masing dokter.

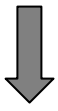
Proses *random* dokter lebih lanjut diuraikan pada lampiran C.

Langkah-langkah dalam proses penugasan dokter adalah sebagai berikut:

1. Cek tanggal penugasan untuk memperoleh *shift* yang masih kosong. Kemudian cek hari minggu dan non-minggu untuk menentukan penggunaan kelompok.
2. Untuk mengisi *shift* yang kosong digunakan proses *greedy* seperti yang telah dijelaskan pada bagian sebelumnya.
3. Lakukan parsing dari *Array of Array of Integer* (S) ke *Array of Integer* (C), sehingga solusi berupa kelompok yang terbentuk akan menjadi kandidat kelompok.

   (S) = {(1,2) (3,4) (1,3) (2,4) (1,4) (2,3) (2,1) (3,2) (4,3) (4,1) (3,1) (4,2)}

   (S) = {(1,2,3) (4,1,2) (3,4,1) (2,3,4) (1,2,4) (3,1,2) (3,4,2) (1,3,4) (1,3,2)
   (2,4,1) (3,1,4) (4,2,3) (1,4,2) (2,3,1) (3,2,4) (4,1,3) (1,4,3) (2,1,4)
   (3,2,1) (2,4,3) (2,1,3) (4,2,1) (4,3,1) (4,3,2)}

    Parsing

   (C) = {(1,2) (3,4) (1,3) (2,4) (1,4) (2,3) (2,1) (3,2) (4,3) (4,1) (3,1) (4,2)}

   (C) = {(1,2,3) (4,1,2) (3,4,1) (2,3,4) (1,2,4) (3,1,2) (3,4,2) (1,3,4) (1,3,2)
   (2,4,1) (3,1,4) (4,2,3) (1,4,2) (2,3,1) (3,2,4) (4,1,3) (1,4,3) (2,1,4)
   (3,2,1) (2,4,3) (2,1,3) (4,2,1) (4,3,1) (4,3,2)}

4. Untuk penugasan pada hari minggu digunakan hasil *random* untuk tiga dokter.

   (C) = {(1,2,3) (4,1,2) (3,4,1) (2,3,4) (1,2,4) (3,1,2) (3,4,2) (1,3,4) (1,3,2)
   (2,4,1) (3,1,4) (4,2,3) (1,4,2) (2,3,1) (3,2,4) (4,1,3) (1,4,3) (2,1,4)

(3,2,1) (2,4,3) (2,1,3) (4,2,1) (4,3,1) (4,3,2)}

Indeks solusinya adalah 5. Karena jumlah hari minggu pada bulan januari 2010 adalah 5 hari, berarti terdapat 15 *shift* jaga yang harus diisi.

(S) = { (..,..,..) (..,..,..) (..,..,..) (..,..,..) (..,..,..) }

Contoh; Untuk mengisi *shift* yang kosong pada hari ke-1, maka ditelusuri kelompok dimulai dari kelompok ke-1 (1,2,3). Uji kelayakan dokter dari kelompok yang dipertimbangkan. Jika dokter layak untuk ditugaskan pada *shift* yang dituju. Maka;

(C) = {(4,1,2) (3,4,1) (2,3,4) (1,2,4) (3,1,2) (3,4,2) (1,3,4) (1,3,2)

(2,4,1) (3,1,4) (4,2,3) (1,4,2) (2,3,1) (3,2,4) (4,1,3) (1,4,3)

(2,1,4) (3,2,1) (2,4,3) (2,1,3) (4,2,1) (4,3,1) (4,3,2)}

(S) = { (1,2,3) (..,..,..) (..,..,..) (..,..,..) (..,..,..) }

Dokter yang diwakili *id Array* 1 memperoleh *shift* pagi, *id Array* 2 memperoleh *shift* sore dan *id Array* 3 memperoleh *shift* malam. Berikut seterusnya sampai tidak terdapat lagi *shift* yang kosong.

5. Untuk penugasan pada hari non-minggu.

(C) = {(1,2) (3,4) (1,3) (2,4) (1,4) (2,3) (2,1) (3,2) (4,3) (4,1) (3,1) (4,2)}

Indeks solusinya adalah 26, karena jumlah hari non-minggu adalah 26 hari berarti 52 *shift* jaga yang harus diisi.

(S) = { (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..)

(..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..)}

Contoh; ditelusuri kelompok ke-1 (1,2). Uji kelayakan dokter dalam kelompok tersebut, ternyata dokter dari *id Array* 2 tidak layak untuk ditugaskan pada *shift*

yang dituju (*shift* malam). Sehingga untuk mengisi *shift* tersebut dilakukan penelusuran dokter dari kelompok lain.

(C) = {(3,4) (1,3) (2,4) (1,4) (2,3) (2,1) (3,2) (4,3) (4,1) (3,1) (4,2)}

(S) = { (1,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..)

(..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..)}

Penelusuran lebih dipermudah karena masing-masing dokter dalam kelompok telah diinisialisasi pada proses sebelumnya. Maka, cukup ditelusuri dokter yang dalam kelompoknya diinisialisasi memperoleh *shift* malam. Ternyata dokter dari kelompok ke-2 (3,4) dengan *id Array* 4 adalah layak. Maka;

(C) = {(1,3) (2,4) (1,4) (2,3) (2,1) (3,2) (4,3) (4,1) (3,1) (4,2)}

(S) = { (1,4) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..)

(..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..) (..,..)}

Dokter yang diwakili *id Array* 1 memperoleh *shift* sore dan *id Array* 4 memperoleh *shift* malam. Berikut seterusnya sampai tidak terdapat lagi *shift* yang kosong.

6. Jika kandidat kelompok (C) telah kosong dan solusi belum terpenuhi dalam arti masih terdapat *shift* yang kosong, maka isi ulang (C) dan lanjutkan penelusuran dari kelompok awal.
7. Untuk memilih dokter yang akan mengisi *shift* yang masih kosong, berlaku *constraints* yang berfungsi sebagai uji kelayakan, yaitu:
    1. Dokter yang memperoleh *shift* malam (M) keesokan hari diberi libur.
    2. Dokter tidak sedang *off*.
    3. Penugasan dokter tidak melewati batas maksimum jaga.

4.1.7.5 Penyelesaian Penjadwalan Dokter Jaga IRNA Utama (Instalasi Rawat Inap Utama) dan IRD (Instalasi Rawat Darurat)

Penyelesaian penjadwalan dokter jaga IRNA Utama dan IRD pada dasarnya adalah sama dengan penjadwalan dokter jaga IRNA. Perbedaannya, IRNA Utama dan IRD menggunakan sistem jaga pagi, sore dan malam setiap harinya (berlaku 3 *shift*). Sedangkan pada IRNA berlaku sistem jaga pagi, sore dan malam untuk hari minggu saja. Maka, proses penghitungan *shift* jaga pada IRNA Utama dan IRD adalah sebagai berikut:

1. Cek jumlah hari dalam 1 bulan, misal 31 hari.

   Pada instalasi ini sistem jaga pada hari minggu dan non minggu adalah sama, sehingga pengecekan jumlah hari minggu dan non-minggu tidak dibutuhkan.
2. Hitung jumlah *shift* yang dibutuhkan selama 1 bulan.

   Total *shift* = 3 x 31 = 93 *shift* (karena berlaku tiga *shift* setiap hari)
3. Hitung nilai maksimum jaga yang dapat diperoleh masing-masing dokter.

   Misalnya ada 10 orang dokter, maka;

   Maksimum jaga = 93/10 = 9 *shift*

   Artinya masing-masing dokter dapat memperoleh maksimal 9 *shift* jaga dalam 1 bulan.

Seperti yang telah dijelaskan, bahwa berlaku tiga *shift* untuk setiap harinya. Oleh sebab itu, proses penugasan dokter cukup menggunakan *random* untuk tiga dokter. Untuk selanjutnya langkah-langkah yang dilakukan adalah sama sehingga tidak perlu diuraikan kembali.

4.2 Perancangan

Pada tahapan ini dilakukan perancangan terhadap sistem berdasarkan analisa permasalahan yang telah dilakukan sebelumnya.

4.2.1 Perancangan Basis Data

Berikut merupakan rancangan basis data untuk melengkapi komponen data sistem.

4.2.1.1 Tabel Dokter

Nama Tabel : tb_dokter

Deskripsi isi : Berisi data dokter.

Volume : -

Primary key : id_dokter

Tabel 4.5 Rancangan Tabel Dokter

Nama Kolom	Tipe Data	*Nullable*	Deskripsi
id_dokter	Text (7)	NOT NULL	ID dokter
nama_dokter	Text (25)	NOT NULL	Nama dokter jaga
unit_kerja	Text (10)	NOT NULL	Nama instalasi penugasan dokter

4.2.1.2 Tabel *Shift*

Nama Tabel : tb_shift

Deskripsi isi : Berisi data *shift* jaga.

Volume : -

Primary key : kode_shift

Tabel 4.6 Rancangan Tabel *Shift*

Nama Kolom	Tipe Data	*Nullable*	Deskripsi
kode_shift	Text (1)	NOT NULL	Kode *shift* jaga
shift	Text (6)	NOT NULL	Nama *shift* jaga
jam	Text (12)	NOT NULL	Jam jaga

4.2.1.3 Tabel *Day Off* Dokter

Nama Tabel : tb_dayoff

Deskripsi isi : Berisi data jadwal *off* dokter.

Volume : -

Primary key : id_dayoff

Tabel 4.7 Rancangan Tabel *Day Off* Dokter

Nama Kolom	Tipe Data	*Nullable*	Deskripsi
id_dayoff	Text (6)	NOT NULL	ID *day off*
id_dokter	Text (7)	NOT NULL	ID dokter
dr_tgl	DateTime	NOT NULL	Tanggal mulai *off*
sampai_tgl	DateTime	NOT NULL	Tanggal akhir *off*
keterangan	Text (15)	NULL	Keterangan

4.2.1.4 Tabel *Request* Jadwal

Nama Tabel : tb_request_jadwal

Deskripsi isi : Berisi data jadwal jaga.

Volume : -

Primary key : id_request

Tabel 4.8 Rancangan Tabel *Request* Jadwal

Nama Kolom	Tipe Data	*Nullable*	Deskripsi
id_request	Text (6)	NOT NULL	ID *request* jaga
id_dokter	Text (7)	NOT NULL	ID dokter
tanggal	DateTime	NOT NULL	Tanggal *request* jadwal
kode_shift	Text (1)	NOT NULL	Kode *shift* jaga

4.2.1.5 Tabel Jadwal Jaga

Nama Tabel : tb_jadwal_jaga

Deskripsi isi : Berisi data jadwal jaga.

Volume : -

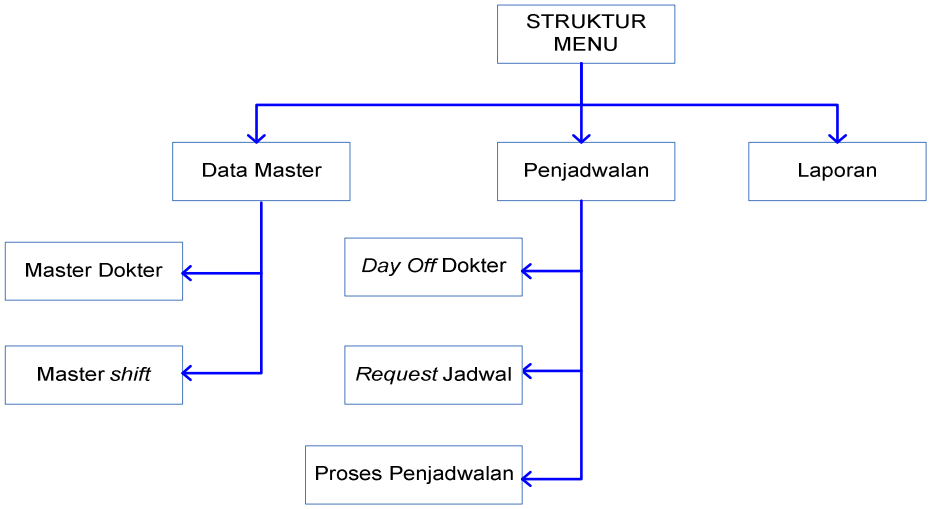
Primary key : id_dokter, tanggal

Tabel 4.9 Rancangan Tabel Jadwal Jaga

Nama Kolom	Tipe Data	*Nullable*	Deskripsi
id_dokter	Text (7)	NOT NULL	ID dokter
tanggal	DateTime	NOT NULL	Tanggal jaga
kode_shift	Text (1)	NOT NULL	Kode *shift* jaga

4.2.2 Perancangan Struktur Menu

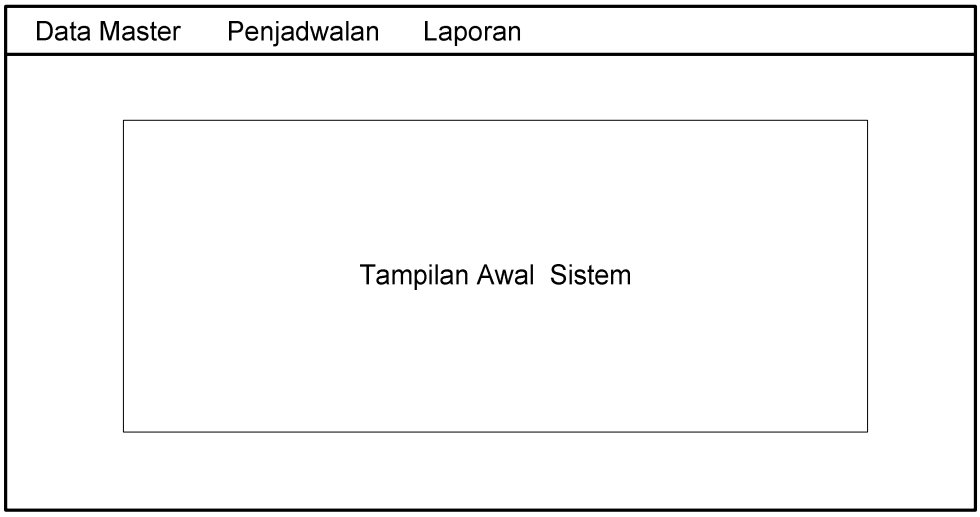
Rancangan struktur menu sistem dapat dilihat dari gambar 4.6.



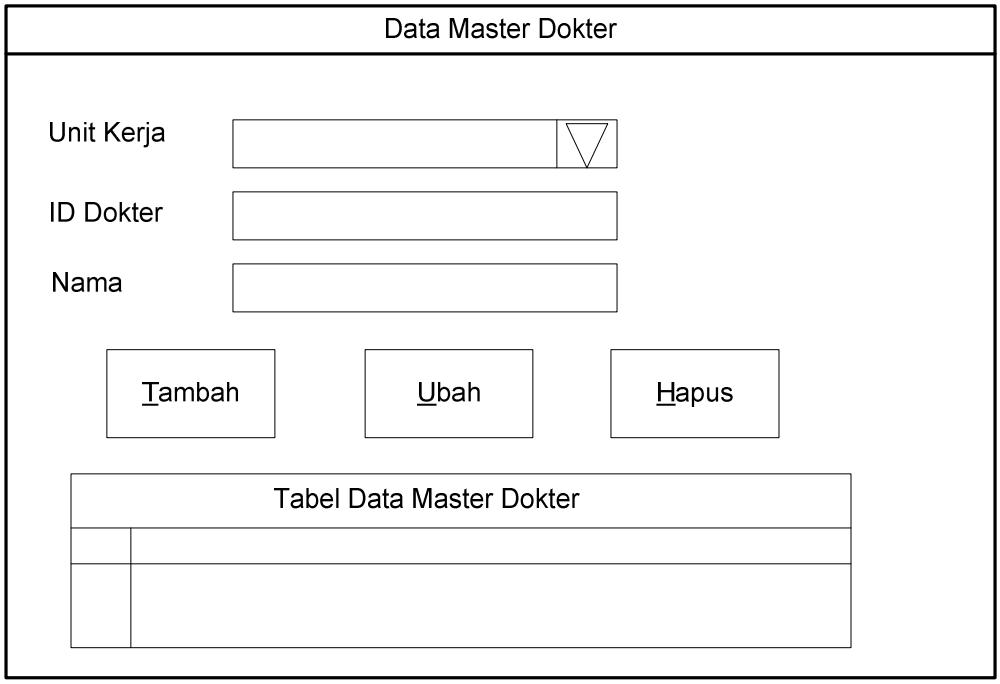
Gambar 4.6 Perancangan Struktur Menu

4.2.3 Perancangan Antar Muka *(Interface)*

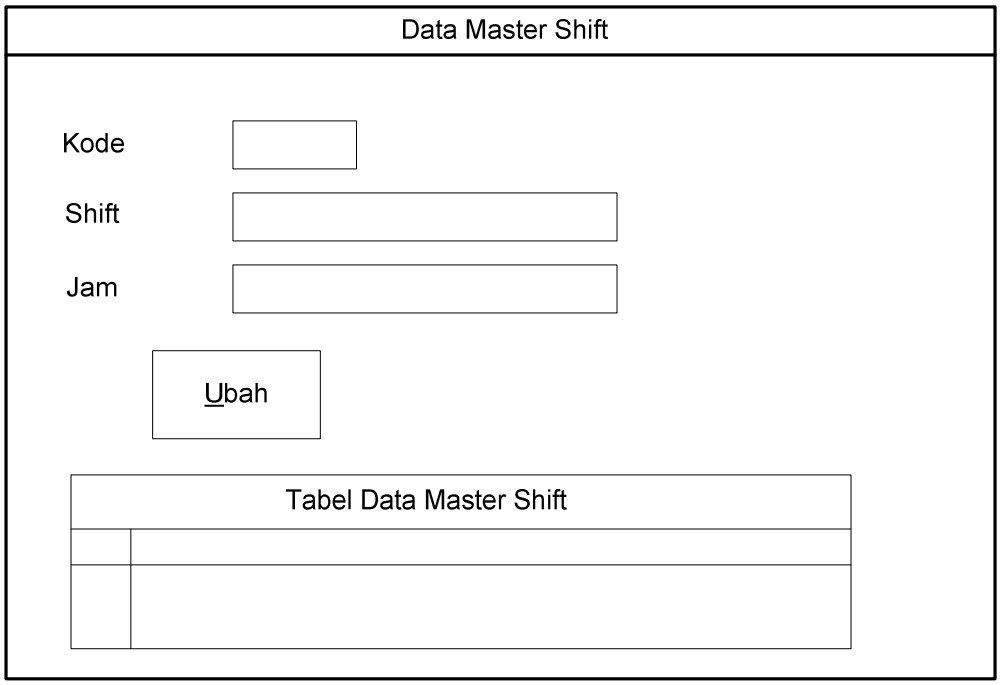
Berikut adalah rancangan antar muka *(interface)* dari sistem yang akan dibangun.



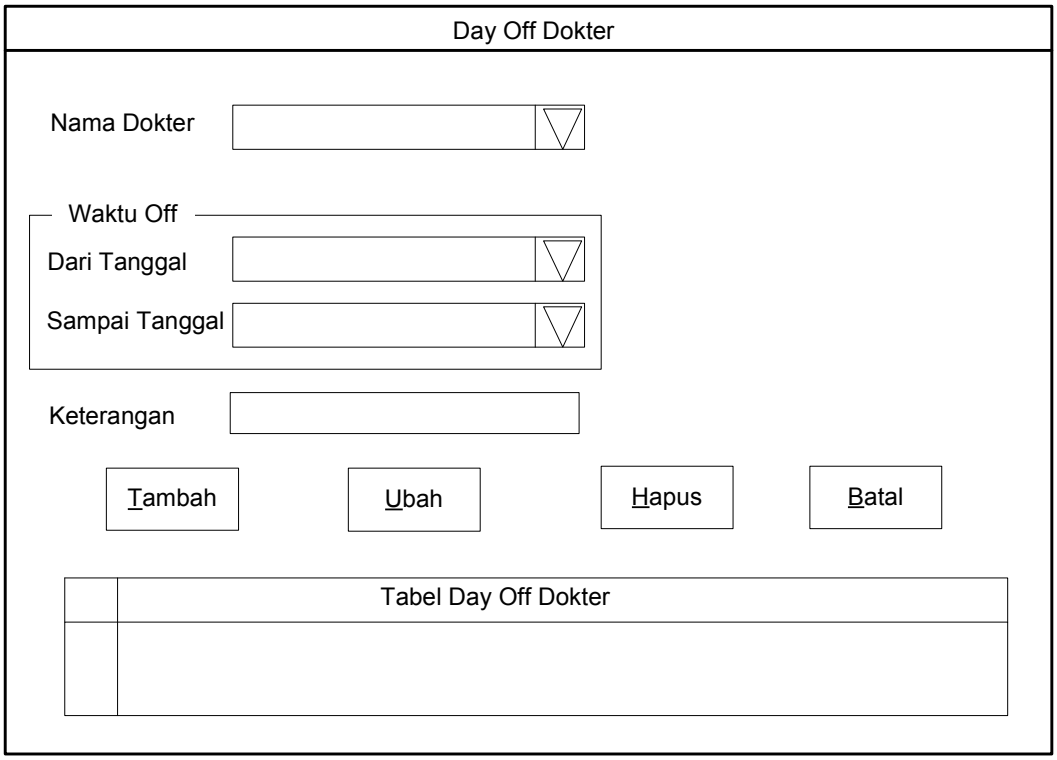
Gambar 4.7 Rancangan *Form* Tampilan Awal Sistem



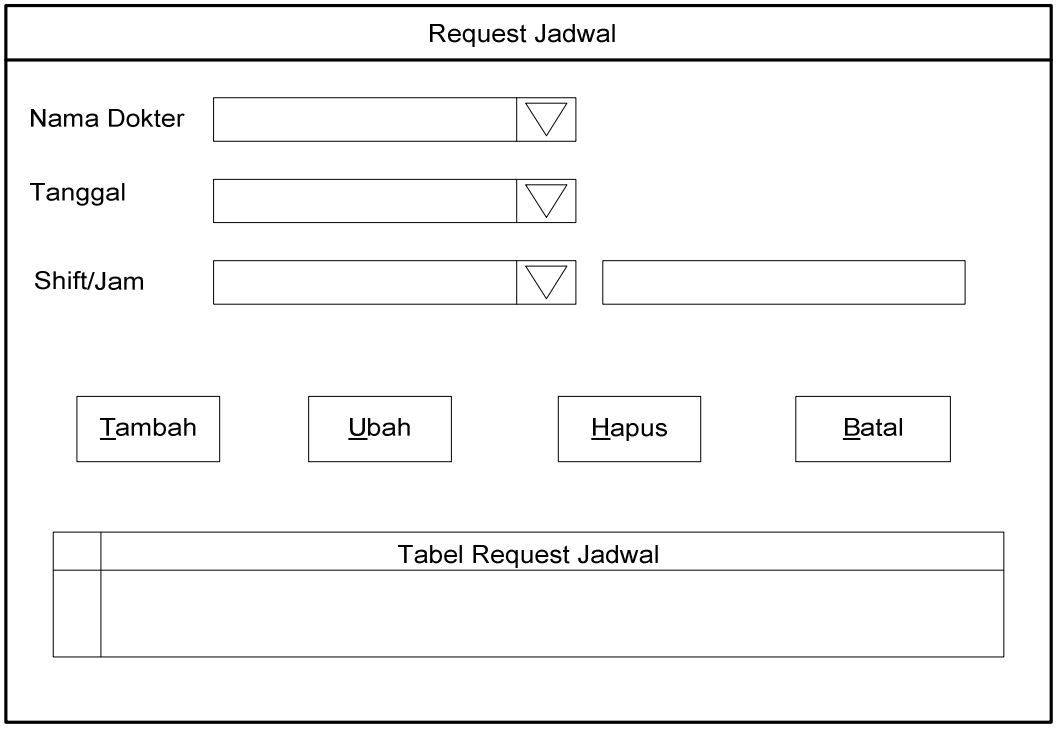
Gambar 4.8 Rancangan *Form* Menu Master Dokter



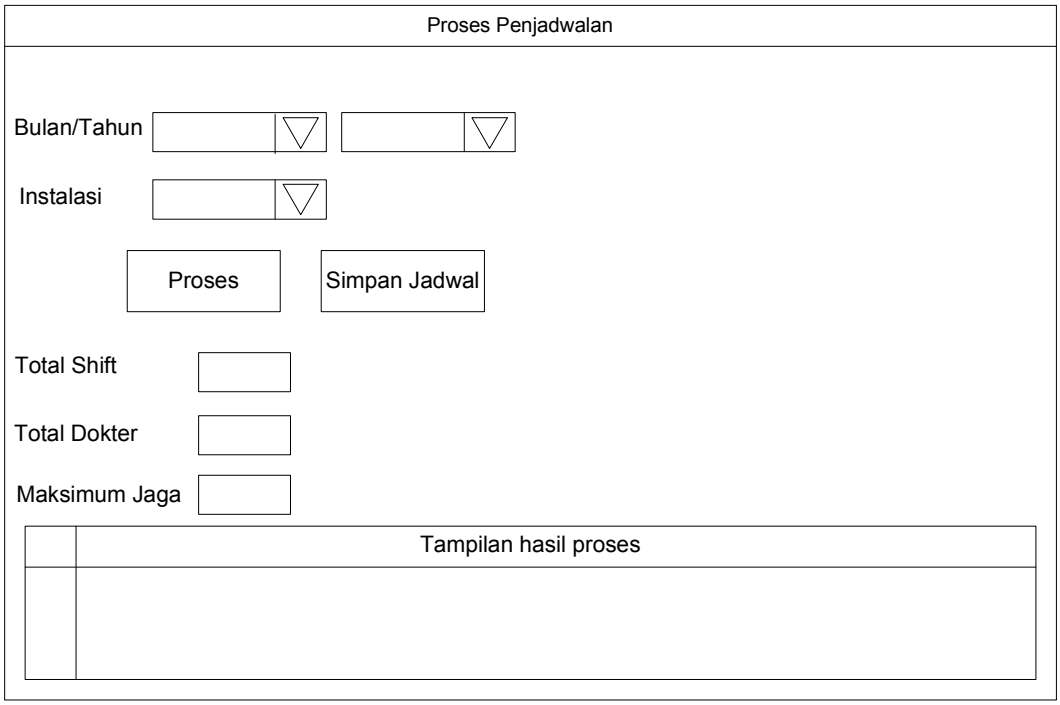
Gambar 4.9 Rancangan *Form* Master *Shift*



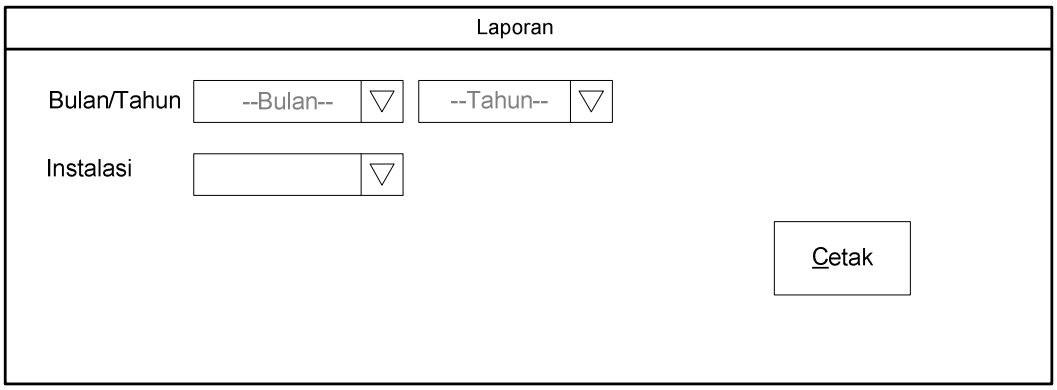
Gambar 4.10 Rancangan *Form* Menu *Day Off* Dokter



Gambar 4.11 Rancangan *Form* Menu *Request* Jadwal



Gambar 4.12 Rancangan *Form* Menu Proses Penjadwalan



Gambar 4.13 Rancangan *Form* Menu Laporan

4.2.4 Rancangan *Procedural*

Berikut ini merupakan *pseudocode* proses penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi yang diambil dari penjadwalan IRNA (Instalasi Rawat Inap).

```
Function PenjadwalanIRNA ()

Deklarasi
Idx  : Indeks_solusi
S2   : Himpunan_solusi2
S3   : Himpunan_solusi3
C    : Kandidat_dokter
x,y,z: dokter
jmldokter : jumlah_dokter
jmlsolusi : jumlah_solusi

Algoritma :
S2 ← {} { inisialisasi S2 dengan kosong }
S3 ← {} { inisialisasi S3 dengan kosong }
For x=1 to jmldokter
   If not solusi (S2) and C ≠ {} then
      x ← seleksi (C) { pilih dokter sebagai x dari solusi }
      C ← C-{x} { dokter dari kandidat berkurang satu }
   Endif
   For y=1 to jmldokter
      If not solusi (S2) and C ≠ {} then
      y ← seleksi (C) { pilih dokter sebagai y dari solusi }
      C ← C-{y} { dokter dari kandidat berkurang satu }
      Endif
        If x ≠ y then  {periksa kelayakan}
        S2 ← S2 ∪ {x,y} {Solusi untuk random 2 dokter}
        Endif
Endfor
Endfor

For x=1 to jmldokter
```

```
   If not solusi (S3) and C ≠ {} then
      x ← seleksi (C) { pilih dokter sebagai x dari solusi }
      C ← C-{x} { dokter dari kandidat berkurang satu }
   Endif
   For y=1 to jmldokter
      If not solusi (S3) and C ≠ {} then
      y ← seleksi (C) { pilih dokter sebagai y dari solusi }
      C ← C-{y} { dokter dari kandidat berkurang satu }
      Endif
      For z=1 to jmldokter
      If not solusi (S3) and C ≠ {} then
      z ← seleksi (C) { pilih dokter sebagai z dari solusi }
      C ← C-{z} { dokter dari kandidat berkurang satu }
      Endif
        If x ≠ y and x ≠ z and y ≠ z  then  {periksa kelayakan}
        S3 ← S3 ∪ {x,y,z} {Solusi untuk random 3 dokter}
      Endif
Endfor
Endfor

For i=1 to jmlsolusi
     ShiftS ← S2(x){inisialisasi dokter dalam kelompok
                    Hasil random 2 dokter}
     ShiftM ← S2(y)
Endfor
        For i=1 to jmlsolusi
           ShiftP  ←  S3(x){inisialisasi  dokter  dalam  kelompok
                           hasil random 3 dokter}
           ShiftS ← S3(y)
           ShiftM ← S3(z)
           Endfor

//pengecekan constraint untuk penugasan
While x <= jmlhari do
  While ShiftM=jadwalM-1 or ShiftM >=maxjaga or ShiftM
  ≠tanggaloff do {Uji kelayakan berdasarkan constraint  malam}
  Idx=Idx + 1
```

```
  If Idx > jmlsolusi then Idx=1
  Endif
  Endwhile
  While ShiftS=jadwalS or ShiftS >=maxjaga or Shifts=jadwalM-1
  Shifts≠tanggaloff do
  Idx=Idx + 1
  If Idx > jmlsolusi then Idx=1 {cek constraint untuk shift sore}
  Endif
  Endwhile

  While ShiftP=jadwalP or ShiftP >=maxjaga or ShiftP=jadwalM-1
  ShiftP≠tanggaloff do
  Idx=Idx + 1
  If Idx > jmlsolusi then Idx=1 {cek constraint untuk shift pagi}
  Endif
  Endwhile

Endwhile

// penghitungan shift
jumlahMinggu=0
For i=1 to jumlahHari
   If hari=sun then
   jumlahMinggu=jumlahMinggu + 1
   Endif
Endfor

jml2shift = (jumlahHari-jumlahMinggu)*2 {jumlah shift untuk hari
                                         dengan 2  shift jaga}

jml3shift = jumlahMinggu*3 {jumlah shift untuk hari dengan 3
                           shift jaga}

totShiftIRNA = jml2shift + jml3shift
totShiftIRNAUtama = jumlahHari*3
totShiftIRD = jumlahHari*3

Maxjaga=totshift/jmldokter {perhitungan maksimum perolehan jaga}
```

BAB V

IMPLEMENTASI DAN PENGUJIAN

Implementasi dan pengujian adalah tahapan yang dilakukan setelah tahap analisa dan perancangan selesai dikerjakan.

5.1 Implementasi

Implementasi merupakan tahap dilakukan pengkodean *(coding)* dan pada tahap ini sistem siap untuk dioperasikan. Sehingga akan diketahui apakah sistem yang dibuat telah mencapai tujuan yang diinginkan.

Sistem penjadwalan dokter jaga menggunakan algoritma greedy dengan permutasi ini dibuat menggunakan perangkat lunak *Microsoft Visual Basic 6.0* dan *Microsoft Access 2003*.

5.1.1 Lingkungan Implementasi

Lingkungan implementasi sistem terdiri atas dua bagian, yaitu; lingkungan perangkat keras dan lingkungan perangkat lunak.

5.1.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan pembuat sistem memiliki spesifikasi sebagai berikut:

1. *Processor* : Intel Core 2 Duo 2.00 GHz.
2. RAM 1 GB dengan *Harddisk* 160 GB

5.1.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan pada implementasi ini adalah sebagai berikut:

1. *Operating System* : *Windows XP Professional*
2. *Microsoft Visual Basic 6.0* dan *Microsoft Access 2003*.

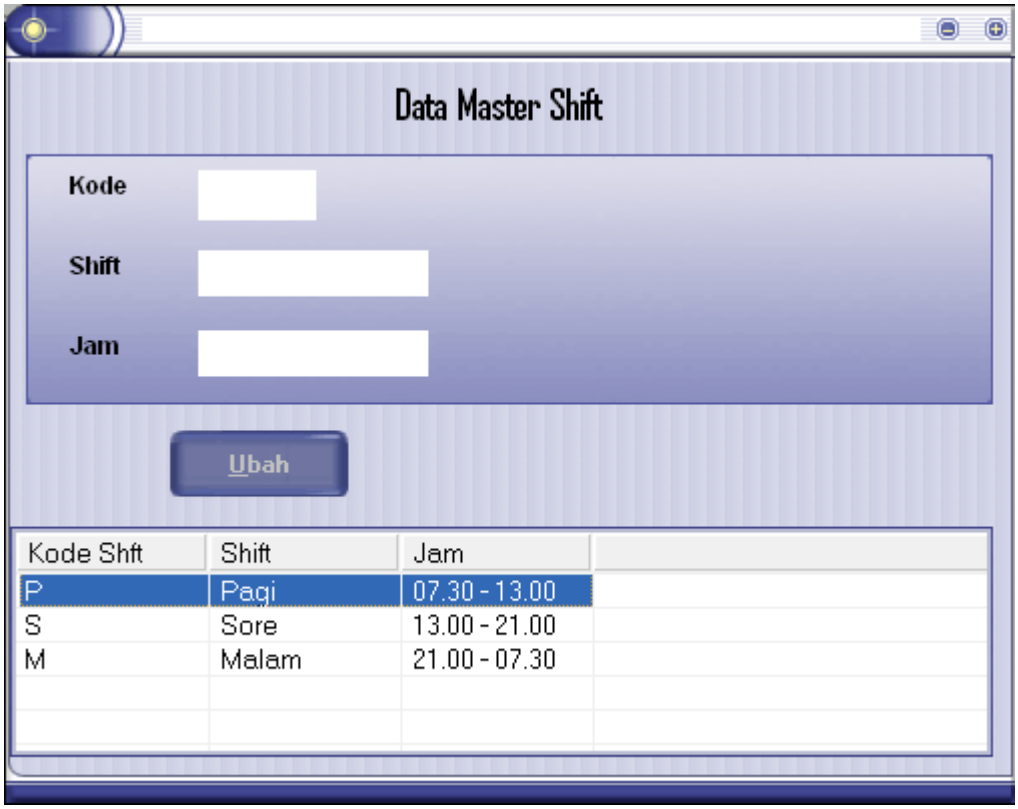
5.1.2 Hasil Implementasi

Hasil implementasi sistem penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi, secara lengkap dapat dilihat dari gambar 5.1 sampai dengan gambar 5.8 berikut ini.

Gambar 5.1 *Form* Utama

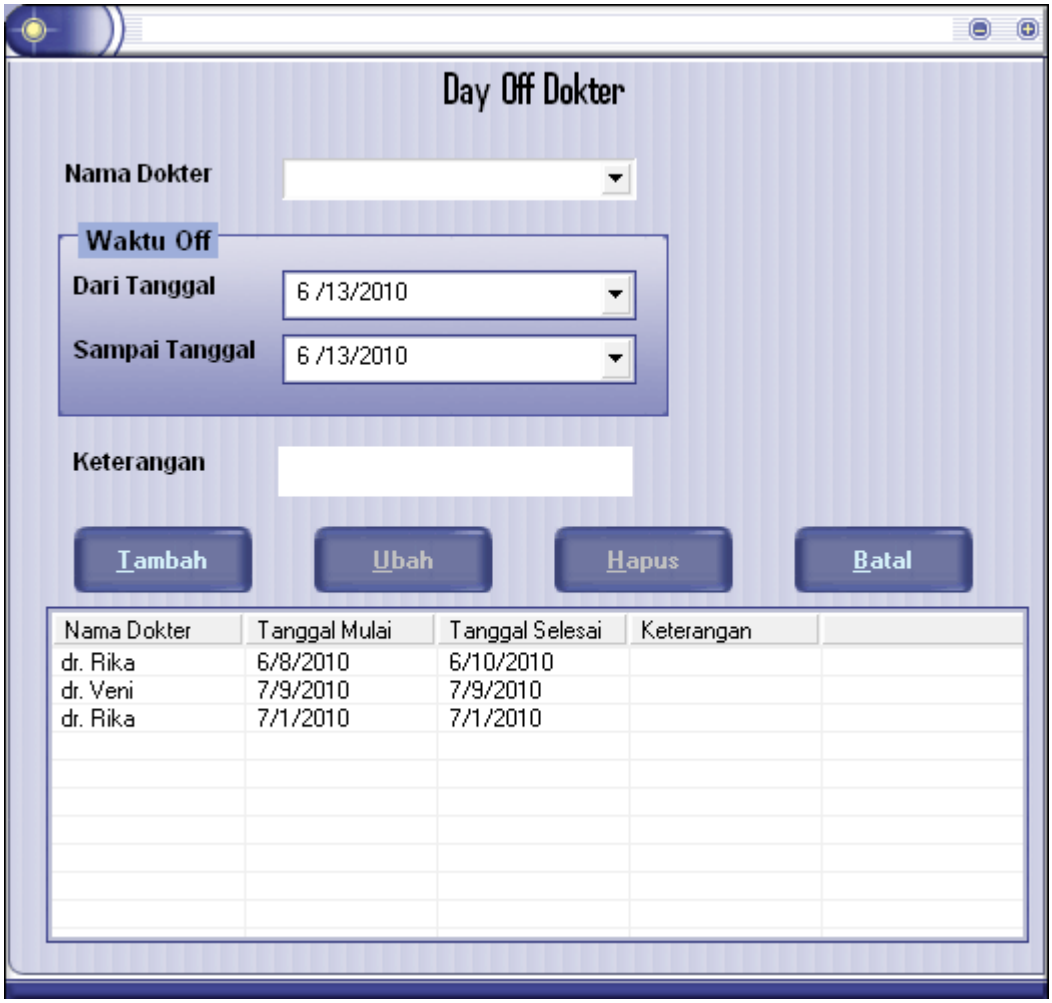
Pada *form* utama terdapat tiga menu utama, yaitu; pengelolaan data master, penjadwalan dan laporan. Proses data master terdiri atas master dokter yang dapat dilihat pada gambar 5.2 dan proses master *shift* dapat dilihat pada gambar 5.3.

Gambar 5.2 *Form* Master Dokter



Gambar 5.3 *Form* Master *Shift*

Pada menu penjadwalan terdapat sub menu, yaitu menu *day off* dokter, *request* jadwal dan proses penjadwalan. Proses *day off* berfungsi untuk mengelola jadwal *off* dokter yang akan menjadi *constraint* pada proses penjadwalan dokter jaga, menu ini dapat dilihat pada gambar 5.4.

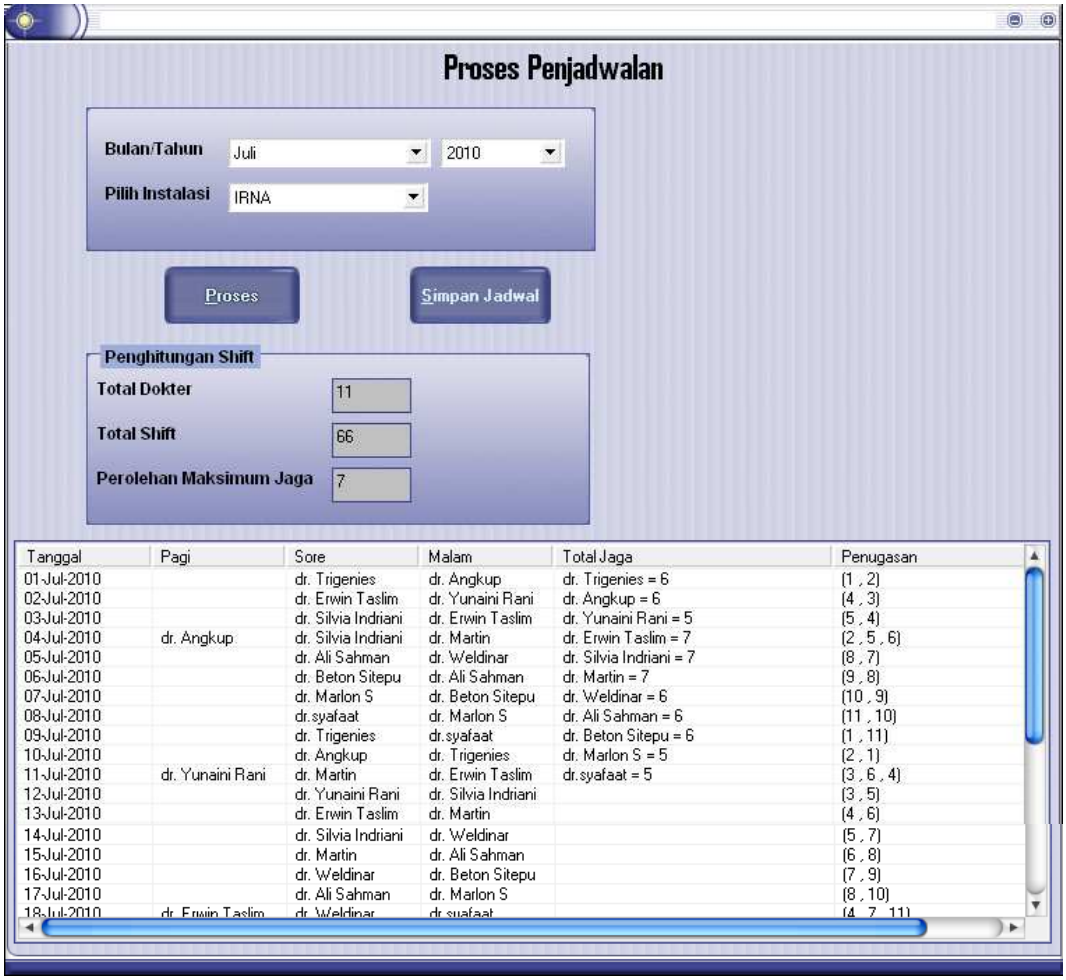


Gambar 5.4 *Form Day Off* Dokter

Sedangkan menu *request* jadwal berfungsi untuk mengelola jadwal jaga berupa tanggal serta *shift* yang diinginkan oleh dokter yang bersangkutan. Penjadwalan dokter jaga dibagi atas tiga instalasi, yaitu; IRNA, IRNA Utama dan IRD. Sistem memberi batasan sesuai *constraint*, dimana form *request* jadwal hanya berlaku pada IRNA Utama dan IRD. Sedangkan pada IRNA dokter tidak diperbolehkan melakukan *request* jaga. Tampilan menu *request* jadwal dapat dilihat dari gambar 5.5 berikut.

Gambar 5.5 *Form Request* Jadwal

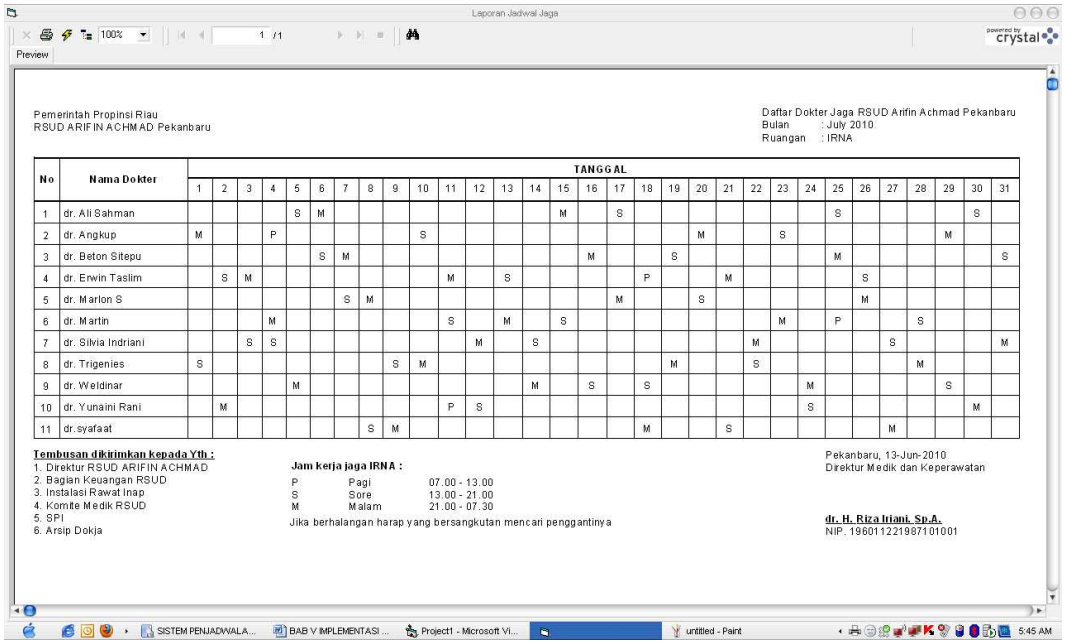
Proses berikutnya dan merupakan proses yang paling utama adalah proses penjadwalan. Proses penjadwalan dilakukan per instalasi sesuai dengan ketentuan *(constraint)* yang ada. Pilih instalasi yang ingin dijadwalkan kemudian *button* proses di klik, maka jadwal jaga akan muncul. Untuk menyimpan jadwal ke tabel jadwal jaga, pengguna harus melakukan klik *button* simpan jadwal. Pada *form* proses juga dapat dilihat perolehan maksimum *shift* jaga masing-masing dokter dalam periode 1 bulan. *Form* proses penjadwalan dapat dilihat pada gambar 5.6.



Gambar 5.6 *Form* Proses Penjadwalan

Menu yang terakhir adalah *report* atau laporan. Laporan dibuat sesuai periode penjadwalan, yaitu berdasarkan bulan/tahun dan instalasi. *Form* laporan jadwal jaga dapat dilihat dari gambar 5.7 dan tampilan hasilnya pada gambar 5.8.

Gambar 5.7 *Form* Laporan Jadwal Jaga



No	Nama Dokter	TANGGAL																											
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	dr. Ali Sahman					S	M									M		S								S					S	
2	dr. Angkup	M			P						S										M			S						M		
3	dr. Beton Sitepu						S	M									M			S						M						S
4	dr. Erwin Taslim		S	M								M		S					P			M					S					
5	dr. Marlon S							S	M									M			S						M					
6	dr. Martin				M						S		M		S									M		P			S			
7	dr. Silvia Indriani			S	S							M		S									M				S					M
8	dr. Trigenies	S								S	M									M			S						M			
9	dr. Weldinar					M									M		S		S						M					S		
10	dr. Yunaini Rani		M									P	S												S						M	
11	dr.syafaat								S	M									M			S						M				

Gambar 5.8 *Form* Hasil Laporan Jadwal Jaga

5.2 Pengujian Sistem

Tahap pengujian dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan analisa dan perancangan serta tujuan yang ingin dicapai. Pengujian juga berfungsi untuk menghasilkan kesimpulan pada tugas akhir.

5.2.1 Identifikasi dan Rencana Pengujian

Identifikasi dan rencana pengujian sistem penjadwalan dokter jaga dapat dilihat pada tabel 5.1 berikut ini.

Tabel 5.1 Identifikasi dan Rencana Pengujian

Modul Pengujian	Butir uji	Tingkat pengujian	Jenis pengujian	Jadwal
Menu-menu yang terdapat pada aplikasi	Normal	Pengujian aplikasi	*Black box*, *User Acceptance Test* (UAT)	disesuaikan

5.2.2 Lingkungan Pengujian

Lingkungan pengujian yang digunakan adalah sebagai berikut:

Perangkat Keras

1. *Processor* : Intel Core 2 Duo
2. *Memory* : 1 GB
3. *Harddisk* : 160 GB

Perangkat Lunak

1. Sistem Operasi : *Windows XP Profesional*
2. Bahasa Pemograman : *Microsoft Visual Basic 6.0*
3. *Database* : *Microsoft Access 2003*

5.2.3 Pengujian Sistem Menggunakan *Black Box*

Pengujian *black box* dilakukan terhadap menu-menu yang tersedia pada aplikasi. Pengujian secara *black box* dapat dilihat pada tabel 5.2 sampai dengan tabel 5.8 berikut ini.

Tabel 5.2 Butir Pengujian *Black Box* Menu Utama

Deskripsi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Kesimpulan
Pengujian menu utama sistem	Klik menu yang diinginkan.	Klik ”data master” dan pilih ”master dokter”	Muncul *form* “master dokter”	Muncul *form* “master dokter”	Berhasil
		Klik ”data master” dan pilih ”master shift”	Muncul *form* “master shift”	Muncul *form* “master shift”	Berhasil
		Klik ”Penjadwalan ” dan pilih ”day off dokter”	Muncul *form* “day off dokter”	Muncul *form* “day off dokter”	Berhasil
		Klik ”Penjadwalan ” dan pilih ”request jadwal”	Muncul *form* “request jadwal”	Muncul *form* “request jadwal”	Berhasil
		Klik ”Penjadwalan	Muncul *form*	Muncul *form*	Berhasil

		” dan pilih ”proses penjadwalan”	“proses penjadwalan ”	“proses penjadwalan ”	
		Klik ”laporan”	Muncul *form* “laporan”	Muncul *form* “laporan”	Berhasil

Tabel 5.3 Butir Pengujian *Black Box* Menu Master Dokter

Deskripsi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Kesimpulan
Pengujian menu master dokter	1. Klik menu ”master dokter”. 2. Klik tombol ”Tambah” untuk menambah kan data dokter. 3. Klik tombol ”Ubah” untuk mengubah data dokter yang telah ada. 4. Klik tombol ”Hapus”	Inputkan seluruh data dokter	Data yang diinputkan dapat dilihat pada *list view* sistem	Data berhasil diinputkan	Berhasil
		Jika ID dokter tidak diisi	Muncul pesan ”ID dokter tidak boleh kosong”	Muncul pesan ”ID dokter tidak boleh kosong”	Berhasil
		Jika nama dokter tidak diisi	Muncul pesan ”Nama tidak boleh kosong”	Muncul pesan ”Nama tidak boleh kosong”	Berhasil
		Jika status dokter tidak diisi	Muncul pesan ”Status	Muncul pesan ”Status	Berhasil

	untuk menghapus data dokter.		tidak boleh kosong”	tidak boleh kosong”	
		Ubah data dokter	Data berhasil diubah	Data berhasil diubah	Berhasil
		Hapus data dokter	Data berhasil dhapus	Data berhasil dhapus	Berhasil
		Input dokter dengan ID yang sama dengan dokter sebelumnya.	Muncul pesan ”ID dokter sudah tersedia”	Muncul pesan ”ID dokter sudah tersedia”	Berhasil

Tabel 5.4 Butir Pengujian *Black Box* Menu Master *Shift*

Deskripsi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Kesimpulan
Pengujian menu master *shift*	1. Klik menu ”master *shift*”. 2. Klik tombol ”Ubah” untuk menambah kan data *shift*.	Ubah data *shift*	Data *shift* berhasil diubah	Data *shift* berhasil diubah	Berhasil

Tabel 5.5 Butir Pengujian *Black Box* Menu *Day Off* Dokter

Deskripsi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Kesimpulan
Pengujian menu *day off* dokter	1. Klik menu ”day off dokter”. 2. Klik tombol ”Tambah” untuk menambah kan data *day off* dokter. 3. Klik tombol ”Ubah” untuk mengubah data *day off* yang telah ada. 4. Klik tombol ”Hapus” untuk menghapus data *day off*. 5. Klik tombol “Batal” untuk membatalkan	Input seluruh data *day off* dokter	Data yang diinputkan dapat dilihat pada *list view* sistem.	Data berhasil diinputkan	Berhasil
		Tambah data *day off*	Data berhasil ditambah	Data berhasil ditambah	Berhasil
		Input tanggal mulai *day off* lebih kecil dari tanggal selesai *day off*	Muncul pesan “Tanggal tidak valid”	Muncul kotak “Tanggal tidak valid”	Berhasil
		Input tanggal dengan bulan yang sama dengan bulan sekarang.	Muncul kotak ”Waktu pengisian day off berakhir”	Muncul kotak ”Waktu pengisian day off berakhir”	Berhasil
		Input tanggal *day off* pada tanggal dokter telah melakukan request jaga.	Muncul kotak “Dokter melakukan request jaga antara	Muncul kotak “Dokter melakukan request jaga antara	Berhasil

	inputan.		tanggal tersebut”	tanggal tersebut”	
		Input tanggal *off* yang sama dengan tanggal input sebelumnya	Muncul kotak ”Dokter masih *off*”	Muncul kotak ”Dokter masih *off*”	Berhasil
		Ubah data *day off*	Data berhasil diubah	Data berhasil diubah	Berhasil
		Hapus data *day off*	Data berhasil dihapus	Data berhasil dihapus	Berhasil

Tabel 5.6 Butir Pengujian *Black Box* Menu *Request* Jadwal

Deskripsi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Kesimpulan
Pengujian menu *request* jadwal	1. Klik menu ”request jadwal”. 2. Klik tombol ”Tambah” untuk menambah kan data *request*	Input seluruh data *request* jaga	Data yang diinputkan dapat dilihat pada *list view* sistem.	Data berhasil diinputkan	Berhasil
		Input tanggal dengan bulan yang sama dengan bulan sekarang.	Muncul pesan ”Waktu Pengisian Request	Muncul pesan ” Waktu Pengisian Request	Berhasil

	jadwal.		Berakhir”	Berakhir”	
	3. Klik tombol ”Ubah” untuk mengubah data *request* jadwal yang telah ada.	Input *request* melebihi maksimum jaga	Muncul pesan ”Dokter telah memenuhi maksimal jaga”	Muncul pesan ”Dokter telah memenuhi maksimal jaga”	Berhasil
	4. Klik tombol ”Hapus” untuk menghapus *request* jadwal. 5. Klik tombol “Batal” untuk membatalkan inputan.	Input *request* pada tanggal dan *shift* yang telah penuh	Muncul pesan ”Request sudah penuh pada tanggal dan shift ini”	Muncul pesan ”Request sudah penuh pada tanggal dan shift ini”	Berhasil
		Input *request* jaga yang lebih dari satu *shift* dalam sehari	Muncul pesan ”Request hanya boleh satu shift dalam satu hari”	Muncul pesan ”Request hanya boleh satu shift dalam satu hari”	Berhasil
		Input *request* pada tanggal *off*	Muncul pesan “Dokter sedang off pada tanggal ini”	Muncul pesan “Dokter sedang off pada tanggal ini”	Berhasil
		Ubah data *request*	Data berhasil	Data berhasil	Berhasil

		jadwal	diubah	diubah	
		Hapus data *request* jadwal	Data berhasil dihapus	Data berhasil dihapus	Berhasil

Tabel 5.7 Butir Pengujian *Black Box* Menu Proses Penjadwalan

Deskripsi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Kesimpulan
Pengujian menu proses penjadwalan	1. Klik menu ”proses penjadwalan” 2. Pilih periode (bulan) yang dijadwalkan. 3. Pilih instalasi. 4. Klik tombol ”Proses” untuk menghasilkan jadwal 5. Klik tombol “Simpan Jadwal” untuk menyimpan jadwal yang	Data dokter jaga dan data *shift*	Berhasil menghasilkan nilai perolehan maksimum jaga	Berhasil menghasilkan nilai perolehan maksimum jaga	Berhasil
			Berhasil menghasilkan jadwal jaga	Berhasil menghasilkan jadwal jaga	Berhasil

	dihasilkan				

Tabel 5.8 Butir Pengujian *Black Box* Menu Laporan

Deskripsi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Kesimpulan
Pengujian menu laporan penjadwalan dokter jaga	1. Klik menu ”laporan” 2. Pilih bulan/tahun yang dijadwalkan. 3. Pilih instalasi. 4. Klik tombol ”Cetak” untuk menghasilkan laporan	Data jadwal jaga	Menghasilkan laporan jadwal dokter jaga	Menghasilkan laporan jadwal dokter jaga	Berhasil

5.2.4 Pengujian Sistem Menggunakan *User Acceptence Test* (UAT)

Pengujian menggunakan *User Acceptence Test* (UAT) adalah jenis pengujian yang dilakukan dengan membuat kuesioner atau angket yang berisi pertanyaan-pertanyaan mengenai tugas akhir. Angket disertai nama, jabatan dan tanda tangan responden (pihak yang mengisi angket). Dalam angket tersebut terdapat sepuluh pertanyaan yang berbentuk *obyektif*, dimana responden dapat memilih jawaban yang sesuai dengan situasi dan kondisi yang sedang dihadapi.

Angket hanya dapat diisi oleh dokter atau pihak yang melakukan penjadwalan dokter jaga.

5.2.4.1 Hasil *User Acceptence Test*

User Acceptence Test melalui pengisian kuesioner menghasilkan kesimpulan bahwa sistem penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi lebih efektif dan efisien bila dibandingkan dengan sistem manual. Aplikasi ini sudah memenuhi standar dan sesuai dengan permasalahan yang dihadapi. Responden juga menyatakan bahwa mereka membutuhkan *training* terlebih dahulu sebelum menggunakan sistem ini, agar mereka mengerti secara detail mengenai aplikasi ini. Kuesioner yang diajukan dapat dilihat pada lampiran D.

Berikut ini adalah jawaban dari kuesioner yang telah disebarkan dan memperlihatkan jumlah responden yang memilih jawaban tersebut.

Tabel 5.9 Jawaban Pertanyaan I Hasil Pengujian Dengan Kuesioner

No	Pertanyaan	Jawaban		
		a	b	c
1	Berapa lama biasanya waktu yang anda perlukan untuk penyusunan jadwal dokter jaga menggunakan sistem manual?		1	2
2	Apakah sering terjadi masalah pada saat penyusunan jadwal jaga?		3	
3	Apakah menurut anda perlu diadakan aplikasi untuk membantu penyusunan jadwal dokter jaga?		3	
4	Bagaimana menurut anda kegiatan jadwal jaga setelah menggunakan aplikasi ini?		3	

5	Menurut anda, perlukah anda menjalankan *training* atau latihan sebelum menggunakan aplikasi ini?		3	

Tabel 5.10 Jawaban Pertanyaan II Hasil Pengujian Dengan Kuesioner

No	**Pertanyaan**	**Jawaban**	
		Ya	**Tidak**
1	Apakah sistem penjadwalan dokter jaga saat ini sudah efektif?		3
2	Apakah aplikasi yang dibuat sudah memenuhi kriteria dan mudah digunakan?	3	
3	Apakah aplikasi yang telah dibuat sudah memenuhi standar dan sesuai dengan permasalahan yang anda hadapi?	3	
4	Apakah penggunaan Sistem Penjadwalan Dokter Jaga lebih cepat dibandingkan dengan sistem lama (manual)?	3	
5	Apakah pekerjaan anda menjadi lebih maksimal setelah menggunakan aplikasi ini?	3	

5.3 Kesimpulan Pengujian

Kesimpulan yang diperoleh dari pengujian sistem ini adalah sebagai berikut:

1. Seluruh menu dan *button* pada sistem penjadwalan dokter jaga berfungsi dengan baik.
2. Penggunaan sistem ini menghasilkan jadwal dengan waktu yang relatif lebih cepat bila dibandingkan dengan penjadwalan secara manual dan hasilnya cukup memuaskan.

3. Sistem ini cukup membantu pengguna dan memaksimalkan kegiatan penyusunan jadwal dokter jaga di RSUD Arifin Achmad Pekanbaru.

BAB VI

PENUTUP

Penutup terdiri dari kesimpulan dan saran yang dapat dilihat pada uraian berikut:

6.1 Kesimpulan

Berdasarkan pembahasan dari bab-bab sebelumnya, maka dapat diperoleh kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian *User Acceptence Test* (UAT) sistem penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi telah sesuai dengan prosedur penjadwalan dokter jaga yang terdapat di RSUD Arifin Achmad Pekanbaru.
2. Pada sistem ini algoritma *greedy* diterapkan untuk menghasilkan kelompok dokter yang akan ditelusuri pada tahap penugasan dokter jaga. Kemungkinan solusi kelompok yang dihasilkan cukup banyak sehingga diperlukan bantuan teori permutasi yang dapat mempermudah penelusuran kelompok untuk menghasilkan solusi penugasan dokter jaga.
3. Sistem penjadwalan dokter jaga memperbolehkan dokter untuk melakukan permintaan *(request)* terhadap jadwal jaga. Namun *request* hanya bisa dilakukan sebelum memasuki periode dari bulan yang dijadwalkan.

4. Sistem penjadwalan dokter jaga dapat menghasilkan laporan jadwal jaga berbentuk *softfile* dalam format *Ms Word*, *Ms Excel*, dan *Acrobat Format* (PDF).
5. Sistem ini belum dapat meng-*handle* persoalan jika terdapat dokter yang berhalangan hadir.

6.2 Saran

Seperti yang telah diuraikan pada kesimpulan bahwa sistem penjadwalan dokter jaga menggunakan algoritma *greedy* dengan permutasi belum dapat menangani persoalan apabila terdapat dokter yang berhalangan hadir. Untuk itu dibutuhkan algoritma lain yang dapat menghasilkan sistem yang mampu meng-*handle* persoalan tersebut.

DAFTAR PUSTAKA

Dossey, Otto, Spence, Vanden. *Discrete Mathematics*. 5th edition. Pearson Education, inc. 2006.

Hendarto, Janoe. *Modul Pengantar Analisis Algoritma*. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada. 2007.

Hidayat Santoso, dan Tommy. *Aplikasi Algoritma Greedy Untuk Mengoptimalkan Efisiensi Pekerjaan*. Jurusan Teknik Informatika, Institut Teknologi Bandung. 2008.

Munir, Rinaldi. 2005. *Diktat Kuliah IF 2251 Strategi Algoritmik.* Bandung: Laboratorium Ilmu dan Rekayasa Komputasi, Departemen Teknik Informatika ITB.

Pantas Panggabean, Henry. *Penjadwalan Job Shop Statik Dengan Algoritma Simulated Annealing*. 2002.

Pressman, Roger S. *Rekayasa Perangkat Lunak.* Andi: Yogyakarta. 2002

Supardi, Yuniar. *Microsoft Visual Basic 6.0 Untuk Segala Tingkat.* PT Elex Media Komputindo : Jakarta. 2006.

Wardana, Ari. *Algoritma Greedy yang Dimodifikasi dalam Pencarian Lintasan Terpendek.* Jurusan Teknik Informatika, Institut Teknologi Bandung. 2008.

William. *Pertandingan Liga Dengan Algoritma Greedy*. Institut Teknologi Bandung. 2008.

Zainudin Zukhri, Shiddiq Alhakim. *Algoritma Semut Pada Penjadwalan Produksi Job Shop*. Jurusan Teknik Informatika, Universitas Islam Indonesia. 2004.

Emira, N. “Wawancara mengenai Penjadwalan Dokter Jaga IRNA Utama”, Dokter Umum RSUD Arifin Achmad, wawancara dilakukan 29 Juli 2009.

Sofirira, Yustisa.H. “Wawancara mengenai Penjadwalan Dokter Jaga IRD”, Dokter Umum RSUD Arifin Achmad, wawancara dilakukan 16 Oktober 2009.

Taslim, Erwin. “Wawancara mengenai Penjadwalan Dokter Jaga IRNA”, Kepala SMF Dokter Umum, wawancara dilakukan 24 Juli 2009.